Title of thesis	**Scheduling Multiproduct Chemical Batch Processes using Matrix Representation**

I, AMIR SHAFEEQ, hereby allow my thesis to be placed at the Information Resource Center (IRC) of Universiti Teknologi PETRONAS (UTP) with the following conditions:

1. The thesis becomes the property of UTP.
2. The IRC of UTP may make copies of the thesis for academic purposes only.
3. This thesis is classified as

[ ] Confidential

[X] Non-confidential

If this thesis is confidential, please state the reason:

____

____

____

The contents of the thesis will remain confidential for __________ years.

Remarks on disclosure:

____

____

____

Endorsed by

[illegible]
Signature of Author

[illegible]
Signature of Supervisor

Permanent Address: House No. 41, Mamdot Block, Mustafa Town, Wahdat Road, Lahore, Punjab, Pakistan

Name of Supervisor
Assoc. Prof. Dr. M.I Abdul Mutalib

Date: 22nd October, 2008

Date: 22nd October, 2008

UNIVERSITI TEKNOLOGI PETRONAS

Approval by Supervisor

The undersigned certify that they have read, and recommend to The Postgraduate Studies Programme for acceptance, a thesis entitled

Scheduling Multiproduct Chemical Batch Processes using Matrix Representation

submitted by

Amir Shafeeq

for the fulfilment of the requirements for the degree of

Doctor of Philosophy in Chemical Engineering

Date: 22nd October, 2008

Signature	:	[illegible]
Main Supervisor	:	Assoc. Prof. Dr. M.I.Abdul Mutalib
Date	:	22nd October, 2008
Co-Supervisor	:	Assoc. Prof. Dr. K.A. Amminudin

UNIVERSITI TEKNOLOGI PETRONAS

Scheduling Multiproduct Chemical Batch Processes using Matrix Representation

By

Amir Shafeeq

A THESIS

SUBMITTED TO THE POSTGRADUATE STUDIES PROGRAMME

AS A REQUIREMENT FOR THE

DEGREE OF DOCTOR OF PHILOSOPHY IN CHEMICAL ENGINEERING

Chemical Engineering

BANDAR SERI ISKANDAR,

PERAK

OCTOBER, 2008

DECLARATION

I hereby declare that the thesis is based on my original work except for quotations and citations which have been duly acknowledged. I also declare that it has not been previously or concurrently submitted for any other degree at UTP or other institutions.

Signature: [illegible]

Name : Amir Shafeeq

Date : 22nd October, 2008

ACKNOWLEDGEMENT

First, I would like to express heartiest gratitude to my supervisor Assoc. Prof. Dr. M.I. Abdul Mutalib for his kind, supportive and encouraging attitude during course of my research. Throughout my research, he provided invaluable guidance, skilled advice and especially a very good company. His guidance provided me with much confidence to accomplish my research successfully. I would also like to appreciate my co-supervisor Assoc. Prof. Dr. K.A. Amminudin for his assistance and guidance during course of my research.

I would like to express my appreciation Ms Haslina Noor Hasni, Ms Norma Abdul Rahim, Mr M. Fadil M. Ariff and Ms Kamaliah Mohd from the office of postgraduate studies for their invaluable contribution and kind assistance in all of my activities which are directly or indirectly related to my research.

I feel great honour to express my thanks to my entire family especially my loving parents, brother, sisters, my beloved wife and my daughter for all their support and motivation.

My special gratitude to my friends and colleagues, Rizwan ul Haq, Faizan Javed, Muhammad Nadeem, Boshara Merghawani and most of all Ayyaz Muhammad for providing me with valuable suggestions and recommendations during the course of my research.

ABSTRACT

Batch process plants are usually designed for the production of specialty and fine chemicals such as paint, food and pharmaceutical to meet specific product requirements as set by current market demand. Batch process plants can be operated as single product in which only one product is produced and multiple products which allow production of more than one product using same batch facility. The economics of the batch process heavily depends on efficient scheduling of the different tasks involved in manufacturing the range of products. The main objective of scheduling is generally to minimize completion time known as the makespan of the batch process. Product sequencing, which is used to set order of products to be produced, has a direct impact on the makespan particularly in the multiple products case. Another effect on makespan is observed for different transfer policies used to transfer the product intermediates between process stages. The generally adopted intermediate transfer policies are (i) zero wait (ZW), (ii) no intermediate storage (NIS), (iii) unlimited intermediate storage (UIS) and (iv) finite intermediate storage (FIS). In the past, the determination of makespan for each transfer policy has been done using a number of mathematical and heuristics approaches. Although these approaches are very efficient and are currently being applied in many chemical process industries but most of them end up with the solution in terms of complex mathematical models that usually lack user interactions for having insights of the scheduling procedure. This motivated the current work to develop relatively simple and interactive alternate approaches to determine makespan. The proposed approach uses matrix to represent the batch process recipe. The matrix is then solved to determine the makespan of a selected production sequence. Rearrangement of the matrix rows according to the varied production sequences possible for the specified batch process recipes enables the makespan to be determined for each sequence. Designer is then provided with the production sequence options with its corresponding makespan from which a selection could be made according to the process requirements.

ABSTRAK

Loji pemprosesan berkelompok selalunya di rekabentuk bagi pembuatan produk kimia khusus seperti cat, makanan dan ubatan untuk memenuhi permintaan pasaran semasa. Ianya boleh dikendalikan bagi penghasilan hanya satu atau pelbagai produk menggunakan kemudahan loji yang sama. Dari segi aspek ekonomi nya, ia sangat bergantung kepada kecekapan di dalam penjadualan pelbagai proses yang terlibat bagi menghasilkan pembuatan produk produk yang di kehendaki. Objektif utama penjadualan pelbagai proses ini adalah untuk mengurangkan julat masa proses bagi operasi pembuatan kesemua produk yang ditetapkan mengikut kuantiti permintaan. Urutan produk yang menentukan susunan penghasilan produk mempunyai kesan terus kepada julat masa proses terutama nya bagi pembuatan produk pelbagai. Suatu lagi aspek yang boleh mempengaruhi julat masa proses adalah polisi pemindahan bahan yang di gunakan untuk memindahkan bahan-bahan yang tehasil dari setiap satu proses ke proses selanjutnya. Secara umumnya, polisi pemindahan bahan yang di gunakan terdiri daripada (i) penantian sifar (ZW), (ii) tanpa simpanan sementara (NIS), (iii) simpanan sementara tanpa had (UIS) dan (iv) simpanan sementara berkongsi (FIS). Terdahulu, kaedah yang digunakan bagi penentuan julat masa proses untuk setiap polisi pemindahan bahan yang digunakan adalah berasaskan kepada pendekatan matematik dan heuristik. Walaupun kesemua pendekatan ini sangat efisien and diguna pakai oleh pihak industri, kaedah yang di guna pakai memerlukan kepada penggunaan model matematik yang kompleks dan selalunya kurang kebolehan berinteraksi dengan pengguna terutamanya di dalam memberi penyelesaian menyeluruh bagi membolehkan pengguna memahami dengan lebih mendalam pemasalahan yang di hadapi selain memberi pilihan di dalam penyelesaiannya. Faktor ini telah menjadi dorongan bagi penyelidikan yang dijalankan dengan bertujuan menghasilkan suatu kaedah yang lebih mudah dan berkebolehan untuk lebih berinteraksi dengan pengguna di dalam menentukan julat masa proses yang optima. Kaedah yang di perkenalkan hanya memerlukan pengguna memberikan maklumat resipi proses bagi penghasilan pelbagai produk yang ditetapkan didalam satu susunan matrik. Matrik ini seterusnya diselesaikan bagi menentukan julat masa proses bagi urutan produk yang di wakili oleh susunan matrik tersebut. Penyusunan semula barisan di dalam matrik megikut

urutan produk seterusnya membolehkan penentuan julat masa proses di buat bagi urutan produk yang lain. Pereka bentuk proses kemudiannya di berikan maklumat urutan produk dan julat masa prosesnya bagi pemilihan di buat ke atas urutan produk yang sesuai bergantung kepada ketetapan proses.

TABLE OF CONTENT

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

MILP	Mixed Integer Linear Programming
MINLP	Mixed Integer Non Linear Programming
ZW	Zero Wait
LW	Limited Wait
UW	Unlimited Wait
NIS	No Intermediate Storage
UIS	Unlimited Intermediate Storage
FIS	Finite Intermediate Storage
LIS	Limited Intermediate Storage
CIS	Common Intermediate Storage
MIS	Mixed Intermediate Storage

LIST OF SYMBOLS

i	rows
j	columns
n	products
m	stages
M	matrix
V	idle time
I	holding time
W	waiting time
T	transfer time
U	stage setup time
H	idle time + stage setup time
G	storage setup time

CHAPTER 1

INTRODUCTION

1.1 General Background

The selection of processing technology for the production of chemicals is usually done with the aim that it must be economical, safe and environmentally friendly. Continuous processes have gained considerable popularity in the middle of twentieth century due to economies of scale and the production of bulk chemicals. Batch processes are receiving considerable attention due to their flexibility for the production of small-volume, high-value added products, to cope with the current changes in market demand. They are generally preferred in the industries producing pharmaceutical, polymer and food product due to its flexibility to accommodate varying production requirements using the same process facilities. The major drawback of batch process industries may be due to the additional costs incurred in terms of labor and time taken for batch feeding, transferring and emptying operations for every new batch of products. However, new control techniques for batch operations have considerably reduced the labor and time requirement by virtue of computer operated plant equipments (Orcun et al. 2001).

Batch processing often require multiple operations, such as mixing, blending, reaction, separation and others. These types of processes are usually arranged in sequence of process stages. The productivity of a batch plant can be increased by reducing the batch process time known as makespan. This is done by minimizing the idle time of each process stage through efficient scheduling. The scheduling for batch processes involves parameters such as process sequencing, transfer policies applied and the use of intermediate storage. The transfer policies adopted normally depends on the type of material being produced and the availability of intermediate storage. Intermediate storage is used to hold product intermediate to reduce the idle time and free the process stage to process another batch thus improving equipment utilization.

The concept of pipeless batch plant has been introduced where material processing takes place in a number of fixed processing stations, and materials are transferred from one processing stage to the other using moveable vessels. At times the same vessel is used to transfer materials as well as to hold the processed materials. Pipeless batch plants have been built and used to produce a number of products such as lubricant oils, paints and inks (Vecchietti and Montagna 1998).

Batch processes are generally categorized as single product and multiple products. Single product batch process refers to the production of only one type of product in repetition while multiple products batch process offers production of different products using the same batch plant facility. Multiple products batch process could be further classified as either Multiproduct or Multipurpose. In multiproduct batch process, all the products follow the same operation sequence. In the case of multipurpose batch process, the products need not follow the same operation sequence and also not necessarily utilizing all the processing stages (Birewar et al. 1997, Smith 2005).

The manufacturing of varying quantities of specialized products poses another challenging task for batch process scheduling and operation. This changes the production scale for the various products in a batch plant to meet the fluctuating market demand. This necessitates an efficient production schedule that controls the sequence and timing of different operations to produce different products. It has forced industry to make effective utilization of available resources using proper scheduling. The scheduling can be done either manually or using computer aided tools. Production scheduling manually is time consuming especially for large range of products and cannot meet the requirement of dynamic market changes. As a result, manual approach has been replaced in the scheduling of many batch processes by computer-aided tools.

The general parameters for batch scheduling normally consists of product sequencing i.e. the order of producing different products using the same batch facility, intermediate transfer policies adopted, transfer and setup time between process stages and the overall structure of processing network for the production of specific products. The efficiency of a batch process schedule is usually measured using the makespan which is the completion time of the batch process to complete all the jobs. The best production schedule is usually the one which offers the least makespan. The makespan may vary depending on the intermediate transfer policies adopted to transfer product intermediates from one stage to the next. These transfer policies are classified as zero wait (ZW), limited wait (LW), unlimited wait (UW) and mixed wait (MW) (Pitty and Karimi, 2008).

Each of the stated transfer policy above is governed by certain rules as suggested in the published literature;

- ZW is strict in a sense that it forces the product intermediate to be transferred immediately as soon as it is produced to the next stage. This transfer policy normally results in the longest makespan. This is because the processing of the next product intermediate in the respective stage is delayed until the processing of the previous product has been completed (Jung et al. 1994, Kim et al.1996, Pitty and Karimi, 2008).

- LW, contrary to ZW, offers some flexibility and allows the product intermediate to reside within the same stage until the availability of the next process stage. However, the nature of the product is such that it can only be held for a limited time period. (Pitty and Karimi, 2008).

- UW, in comparison to ZW and LW, offers more flexibility. This policy does not offer any restriction on the time period for storing the product intermediate in case the next stage is not available. UW is further categorized according to the storage configuration used i.e. NIS, UIS, LIS and MIS (Pitty and Karimi, 2008).

    - No intermediate storage (NIS) configuration allows the product intermediate to stay within the same stage if the next stage is not available (Pitty and Karimi, 2008).

- In unlimited intermediate storage (UIS) configuration, temporary storage tanks are used between the process stages to hold the products intermediates till the next stage becomes available. In UIS, temporary storage tanks are always made available between any two stages to free the processing stage to process as many products as possible without the need to worry about where the product intermediate can be stored (Pitty and Karimi, 2008).

- Contrary to UIS, the number of temporary storages is limited in limited intermediate storage (LIS). In LIS, it may happen that temporary storage is not available for the next product intermediate because at that point in time, it is still holding the previous product intermediate. LIS is further classified as LIS-D (dedicated) and LIS-S (shared). In LIS-D, the temporary storage between process stages is dedicated i.e. it can not be shared among process stages whereas in LIS-S, the temporary storage can be shared among the process stages. Most of the past literature uses the term FIS (finite intermediate storage) instead of LIS to describe the limited storage configuration. FIS means the numbers of temporary storages between process stages is finite or fixed (Pitty and Karimi, 2008).

- In mixed intermediate storage (MIS), mix of any of the above storage configuration can be used between process stages e.g. there could exist NIS at one stage and UIS at another stage (Kim et al. 1996, Pitty and Karimi, 2008).

- In mixed wait (MW), mix of any of the above transfer policies can be used between process stages e.g. there could exist NIS/UW at one stage and ZW at another stage (Pitty and Karimi, 2008).

1.2 Objectives of Research

The main objective of this research work is to develop a batch process scheduling approach which would determine makespan of a batch process using newly developed algorithms. The research objectives in detail are presented as follows:

To develop simple algorithms which can be computed efficiently for determining the makespan of a batch process using matrix representation for the various intermediate transfer policies. The input to the algorithm consists of the batch process recipes represented in the form of a matrix.

To develop a screening approach that allows process planner interaction for screening the best solutions for the batch process. For this purpose, a partial enumeration technique would be developed using a set of heuristics rules. The purpose of introducing the heuristic rules is to reduce the amount of CPU time in searching for the optimal solution.

To develop a computer code combining the makespan calculation and the screening approach for optimization purpose using a selected programming language which in this case is Microsoft Visual C++ ^{TM}.

To verify and validate the results obtained from the developed method against some case studies available in the literature related to batch process scheduling.

1.3 Scope of Research

The research work focuses towards developing a systematic and simple method for the scheduling of multiproduct batch process. The method combines a set of newly developed algorithms for makespan calculation taking into consideration of the various transfer policies adopted, and a simple screening method which allows process planner's interaction while choosing the best solutions. The developed algorithm takes the batch process recipes represented in the form of a matrix as its input. The input matrix is then solved to determine makespan using some simple mathematical equations developed from careful observation made on the Gantt chart method for determining makespan. The method is applied for determining optimal production sequence with minimum makespan from all the possible production sequences of the given batch process using some proposed heuristic guidelines. The heuristic guidelines are capable to reduce the solution search space from which enumeration is then applied to find the optimal solution. The amount of CPU time used to determine the optimal solution is reduced significantly. For the sake of simplicity, the transfer and setup time are assumed initially to be negligible. However, to incorporate real world industrial applications, some modifications in the newly developed makespan calculation algorithm are proposed to consider the transfer and setup time. The verification and validation of the developed method is done using case studies available in batch scheduling literature with the aid of a computer code developed for this purpose in Microsoft Visual C++ ^{TM}.

1.4 Organization of Thesis

The major objective of Chapter 2 is to describe a general background of the various scheduling techniques available in literature for batch processing. It also highlights the importance of makespan being the most common parameter used as optimal criteria in all the available techniques for determining optimum solution for different types of batch processes.

Chapter 3 describes in detail the types of batch processes available and their modes of operation with respect to process requirement. Types of intermediate transfer policies adopted in literature and their role in batch process scheduling are also discussed.

Chapter 4 elaborates the development of the proposed method for makespan calculation in the case of various transfer policies. The development of the new method involves the representation of the batch process recipe in the form of a matrix which is then subjected to the formulated mathematical equations for makespan determination. In addition, this chapter also highlights the importance of the proposed method for the determination of the idle time between stages for all transfer policies and more specifically the number and location of temporary storages in UIS/UW and FIS/UW transfer policies.

The real world batch scheduling problems must consider the transfer and setup time for the product intermediates between process stages. Chapter 5, thus, deals with such problems and highlights the improvement made in the newly developed makespan calculation algorithms to incorporate the transfer and setup time.

Chapter 6 explains the approach for screening the optimal solution using the newly developed method for various transfer policies as discussed in Chapter 4. The screening for optimal solution for a given batch process is based on the criteria of minimum makespan. The screening approach locates the optimal production sequence with minimum makespan from the list of possible production sequences for a given batch process using a computer code developed for this purpose. Lastly, a heuristic approach is proposed to use the partial enumeration technique instead of complete enumeration to search for the optimal solution with significantly reduced computational time. To verify the proposed matrix procedure, makespan for different batch process recipes taken from case studies available in literature have also been determined. In addition, some more problems of varying size have been solved to show the variation of computational time with problem size.

The significance of the newly developed method with respect to the existing available in literature is discussed in Chapter 7 followed by some recommendations for the future work.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The purpose of the chapter is to provide a brief overview and summary of important and leading contributions in the field of batch process scheduling. The notable difference in the solution strategies proposed by many researchers in this field of interest is the application of various methodologies and algorithms to obtain optimum solution which is the minimum time span for the batch process. Batch process is preferably used in producing low-volume and high-value-added products. The products are produced based on the various recipes and formulations but share the same process facility. This has led to some complexity in terms of scheduling the various tasks involved.

Batch scheduling determines the order of products to be produced according to current market demand with optimum production cost. The types of scheduling algorithms available in literature include mathematical and search methods. Mathematical methods such as mixed-integer linear programming (MILP) and non-linear programming (MINLP) are capable of providing optimal solutions for small and medium size problems. However, the time to reach the optimal solution increases as the problem size grows. For large scale batch scheduling problems, literature recommends heuristics and metaheuristics search methods such as genetic algorithm, simulated annealing and tabu search.

Genetic algorithm works on the principle of producing new generations by mutation and crossover of parent chromosomes. In batch scheduling new generations resembles to the creation of new production sequences from the previous ones by changing the position of the products in a particular sequence. While, Simulated Annealing works on the principle of annealing of metals to produce crystals. The atoms of the metal are dislocated from their original position by heating. The process of slow cooling causes the atoms to rearrange thus forming crystals. In a batch scheduling problem, simulated annealing starts with the initial solution selected randomly or according to some predefined criteria. The other solutions are compared with the initial solution to search for the optimal solution. On the other hand, Tabu search uses a "tabu list" to keep a record of all the possible solutions of a batch scheduling problem that have been searched for the optimum to avoid the method from revisiting these solutions. (Edgar et al. 2001).

In the multiproduct and multipurpose batch process, the scheduling process follows some predefined criteria. The individual importance of each criterion varies with the production requirement. Irrespective of the criterion chosen for scheduling, the final objective is the allocation of total production time for each product in the production sequence. The two generally observed criteria are the makespan and due date.

2.2 Makespan Criteria

The completion time or makespan is defined as the total time to process a batch of products. Minimizing the makespan could allow the next batch of products to be started earlier. The objective of makespan minimization can be achieved by different methods. One of the possible methods is sequencing. Sequencing is defined as the order in which products are manufactured in a batch process (Kuriyan et al. 1987; Kuriyan and Reklaitis, 1989). In any batch process, it is usual for every product to have different processing times in every stage. Also, there are some stages which remain idle before processing the next product in a particular sequence thereby increasing the makespan. The change in the production sequence could result in relocating the position of the products in every stage which results in changing the amount and location of these idle times. A newly generated sequence could offer a lesser makespan than the previous one.

Another method to minimize process makespan in a batch process is the addition of process units in parallel to the existing ones. The strategy behind the use of parallel units is to perform simultaneous unit operations thereby reducing the makespan. Again, the objective is to reduce the idle time of process stages in a particular production sequence. The additional process units could either be identical or non identical to the existing ones (Voudouris and Grossmann, 1992, Smith, 2005).

The transfer of the product intermediates from one stage to the other follows a number of transfer policies as specified in the literature. The makespan of any batch process also depends significantly on the transfer policy specified for transferring the product intermediates between stages (Kim et al. 1996, Bassett et al. 1997).

2.3 Due Date Criteria

The time period during which a production target has to be achieved is known as due date. The due date for the production of any product is assigned as per customer requirement (Cerda et al. 1997). The due date depends on two important parameters. The first parameter, Earliness refers to the production goal which is achieved well before the specified production time. In some cases, this could be costly. This is because the customer refuses to accept the product before his specified due date. This could result in increasing the inventory cost for storing the product till final delivery. The second parameter, Tardiness refers to the production of products after the specified due date. This could result in the cancellation of the next production order from the customer. The due date criteria as per customer requirements can be achieved using the effective scheduling techniques (Ryu et al. 2001). The scheduling techniques could be similar to those stated earlier for meeting the objective of minimum makespan.

The formulation of batch scheduling problems could be done using various methods available in the literature. These methods have been applied for both general types of batch processes i.e. Multiproduct and Multipurpose, and are discussed below.

2.4 Batch Scheduling Methods

The batch scheduling methods involve mathematical programming methods such as mixed integer linear program (MILP) and mixed integer non linear program (MINLP) as well as heuristics and metaheuristics such as genetic algorithm, simulated annealing and tabu search. The applications of these methods to solve batch scheduling problems by a number of researchers are presented below.

2.4.1 Mathematical Programming Methods

2.4.1.1 Multiproduct Batch Process

The work on developing various algorithms for the purpose of makespan determination followed by its minimization in the case of multiproduct batch processes for various intermediate transfer policies have been reported significantly using mathematical programming methods in the past literature. Ku and Karimi (1988) proposed MILP formulations for scheduling of multiproduct batch process with finite intermediate storage (FIS). The use of temporary storage improved the process efficiency by reducing the idle time between process stages. Their formulations considered ZW transfer policy with FIS in cases where temporary storage is not available or when there is an immediate need to transfer the product intermediate to the other stage. The number and location of temporary storages are fixed according to FIS. Their proposed formulations considered the processing times of products only with negligible transfer and setup time. The optimization software used for solving MILP formulations was LINDO (linear interactive discrete optimizer). Later, Rajagopalan and Karimi (1989) developed completion time algorithms for multiproduct batch processes with mixed intermediate storage (MIS) using

ZW/NIS/FIS/UIS. They also included transfer and setup time in the proposed formulations.

Ku and Karimi (1990) extended their work further on FIS by using it in combination with ZW and NIS to work on the shared storage system. The shared storage was introduced to optimize the number of temporary storages required. The proposed algorithms also included the transfer and setup time with processing times for various products. However, Ku and Karimi (1992) presented MILP formulations specifically to determine the optimum production sequence with minimum makespan for the case of no intermediate storage (NIS) in multiproduct batch process. The transfer and setup time were not considered in the formulation. The formulations were solved using GAMS (General Algebraic Modeling Software).

Birewar and Grossmann (1989a,b) incorporated scheduling of multiproduct batch plants by considering one unit per processing stage. They developed MILP formulations to determine the production sequence with minimum makespan for the case of mixed product campaign in ZW and UIS. In addition, they also incorporate the cleanup time in the developed MILP formulations. The formulations were solved using GAMS. On the other hand, Jung et al. (1994) presented MILP formulation to calculate completion time of each product at each stage in a multiproduct batch process with zero wait (ZW) transfer policy. The transfer and setup time in MILP formulations were assumed negligible but were still incorporated in the MINLP formulations. The optimization software used for this purpose was LINDO. Later, Jung et al. (1996) worked on shared temporary storage and developed completion time algorithm in the case of multiproduct batch processes. They worked on developing the common intermediate storage (CIS) using FIS/ZW and NIS/ZW transfer policies. Their model also incorporated the transfer and setup time.

Moon et al. (1996) introduced some new MILP formulations for multiproduct batch process scheduling with ZW transfer policy. Their MILP formulations also incorporate the transfer and setup time and were able to determine the possible production sequence with minimum makespan. They developed various MILP models to solve batch scheduling problems for cases involving single product campaign as well as mixed product campaign. Kim et al. (1996) proposed improved MILP formulations for determining the completion time for ZW, NIS, UIS, FIS and MIS. For the incorporation of transfer and setup time, MINLP formulations were also suggested.

For efficient scheduling operation with temporary storages, the location and storage time of product intermediates in temporary storages must be given priority while making any scheduling decision (Vecchietti and Montagna, 1998). Ha et al. (2000) proposed MILP formulations to find the minimum makespan for multiproduct batch processes with special consideration to storage time of product intermediates in storage tanks. The effect of location and number of intermediate storage tank on scheduling was also considered. Their MILP formulations considered mix of zero wait (ZW), no intermediate storage (NIS) and unlimited intermediate storage (UIS) between process stages. A significant work on the application of temporary storages in the design of batch processes has also been reported (Takeichiro et al. 1982, 1984; Karimi and Reklaitis, 1983; Modi and Karimi, 1989).

The parallel equipments could reduce the inter-stage idle time by simultaneous unit operation thereby lowering the process makespan. The additional equipment is usually installed in parallel to the existing equipment with longest processing time. The parallel equipments could be identical or non identical based on the process requirements. A number of MILP and MINLP formulations have been developed for scheduling batch process with parallel equipments and with transfer and setup time (Birewar and Grossmann, 1990; Hui and Gupta, 2001; Chen et al. 2002; Gupta and Karimi, 2003b; He and Hui, 2006; Liu and Karimi, 2007a,b; 2008). The advantage of parallel equipment in single product is more pronounced compared to multiproduct/multipurpose batch process. This is because in single product batch process, same product is being produced repeatedly and the stage with longest processing times would be the same in every production cycle (Lee and Lee 1996; Ryu and Lee, 1997).

Apart from scheduling, some of the work has been specifically reported for the design of multiproduct batch plants using parallel equipments (Vaselenak et al. 1987). Voudouris and Grossmann (1993) worked on the synthesis of multiproduct batch plants using MINLP. Their work mostly covered the aspects of product sequencing, equipment sizing and optimal allocation of intermediate storage tanks. Ravemark and Rippin (1998) developed MINLP model for the preliminary design of multiproduct batch plants using parallel equipments and also determined the optimal location and size of intermediate storage installed between the process stages.

Batch process scheduling under various uncertainties poses a challenging task. The uncertainties could either be short term or long term and due to many reasons such as uncertainty in product demand, variations in customer requirement or raw material supply (Wellons and Reklaitis, 1989). A number of algorithms have been developed for scheduling multiproduct batch processes under such circumstances (Petkov and Maranas, 1997; 1998a,b; Vin and Ierapetritou, 2000; 2001). In addition, variation in the processing times of different products during production period also sometimes results in uncertainty and it is suggested to perform rescheduling to meet production requirements (Balasubramanian and Grossmann, 2002; Ryu et al. 2007).

Scheduling in multiproduct batch processes with due date penalties has also been a wide area of interest. The due date penalty could be reduced by incorporating the customer satisfaction and with effective batch production sequence in production scheduling. Ku and Karimi (1991a) proposed four different types of algorithms for scheduling serial multiproduct process with a single batch unit in each stage with arbitrary intermediate storage policies. The objective is to minimize the total penalty due to late deliveries to meet customer satisfaction. On the other hand, Kudva et al. (1994) proposed new algorithm considering finite intermediate storage between all stages to meet the order deadlines. The priority assigned to each order was as per due date and product importance.

The MILP formulations were also suggested for other batch processing tasks. These include flexible equipment allocation and variable batch sizes in the case of mixed intermediate storage (MIS) in multiproduct batch processes with the objective to maximize the profit (Kondili et al. 1993, Blomer and Günther, 1998). The controlling parameters in these MILP formulations were the sequencing of products and scheduling of various processing tasks such as equipment allocation for products, cleaning of equipments and use of temporary storages. Gupta and Karimi (2003a) presented a two step MILP method to address the issue of limited shelf-lives of intermediate products, batch splitting at the storage, a batch filling multiple orders and general product specifications in case of multiproduct batch processes. Chan and Hui (2003) introduced a stepwise approach with MILP formulation to schedule single stage multiproduct batch plants. The basic idea behind the stepwise approach was to add new production orders sequentially into the existing schedule. More work on developing MILP models for short term scheduling of multistage, multiproduct batch plants have been reported by Gupta and Karimi (2003b), and, Castro and Grossmann (2005). The proposed MILP models were shown to be efficient for many objective functions such as minimization of total cost, total earliness and makespan.

2.4.1.2 Multipurpose Batch Process

The scheduling task for the case of multipurpose batch process is more complex compared to multiproduct batch process. This is because for the case of multipurpose batch process, the products need not to pass through each stage. Mostly, the scheduling algorithms related to multipurpose batch process are developed using MINLP. This is because of the presence of different processing paths for different products thus causing non linearity among certain variables incorporated in the scheduling algorithms (Suhami and Mah, 1982; Cerda et al. 1989; Henning et al. 1994; Grau et al. 1996). Some of the earlier work on multipurpose batch process was specifically done for scheduling and design in the case of single and mixed production campaigns with intermediate storage (Wellons and Rekalitis 1989a,b; 1991a,b).

The scheduling of multipurpose batch process can either be based on single route or multiple routes. Single route scheduling means the production path for all the products remain the same but all the products does not necessarily pass through each process stage. On the other hand, in multiple route problems, different products follow different production paths in a batch process (Faqir and Karimi, 1989, Kim et al. 2000).

The scheduling objectives for multipurpose batch process are mostly same to those observed earlier in the case of multiproduct batch process. These include makespan minimization and optimal allocation of temporary storages for different intermediate transfer policies. Kiraly et al. (1989) proposed a two stage procedure for optimal design of multipurpose batch plants. The procedure involved generating a set of alternative campaigns for each product and selecting the campaign offering the minimum cost. Patsidou and Kantor (1991) proposed MILP formulations to graphically solve scheduling problems in multi-purpose batch plants, with special consideration given to the various intermediate storage policies. Henning et al. (1994) proposed the scheduling of multipurpose batch plants using proper allocation of intermediate storage. Their procedure considered batch mixing and splitting, fixed processing times and production of the same material using available processing tasks. Voudouris and Grossmann (1996) developed MILP model to integrate designing and scheduling of multipurpose batch plants while targeting the existence of intermediate storage tanks in the production path.

Similar to the case of multiproduct batch process, the use of parallel equipments in multipurpose batch process could possibly reduce the makespan and increase the production capacity (Papageorgaki and Reklaitis 1990a, b; 1993; Pinto and Grossmann 1995). Moon and Hrymak (1999) presented MILP model for scheduling of sequential multipurpose batch plant to determine the optimum movement plan for maximum equipment utilization with various parallel tasks. The objective was to minimize process makespan.

Rodrigues et al. (2000a,b) proposed MILP models for short term scheduling in multipurpose batch plants. Their approach was directed towards the scheduling problem when product demands are as per customer orders. Heo et al. (2003) presented 3 MILP models for scheduling and design of multipurpose batch plants under NIS and ZW transfer policy. The first MILP model gave minimum number of equipment units required to produce the products, the second MILP model determined the minimum cycle time and third MILP model determined the equipment size and scheduling that minimized the cost. Bonfill et al. (2005) addressed the short term scheduling problem in chemical multipurpose multistage batch processes with variable processing times using MILP formulations. The MILP model thus developed provided information on production sequence, assignment of tasks to stages and expected waiting and idle time between process stages.

2.4.2 Heuristics and Metaheuristics

The batch scheduling using heuristic and metaheuristics was proposed to reduce the complexity and limitation of MILP and MINLP methods in large scale problems. Like other scheduling methods, heuristics and metaheuristics were also developed to address various issues in batch scheduling of multiproduct and multipurpose batch processes for different transfer policies.

2.4.2.1 Multiproduct Batch Process

A computer code for short term scheduling for multiproduct batch plants was developed by Mauderli and Rippin (1979), and, Egli and Rippin (1986). They used simple heuristics to consider various aspects such as available equipment, product delivery dates and equipment requirement for each product.

Suhami and Mah (1981) modeled batch process scheduling problem with no intermediate storage (NIS) using branch and bound procedure. The objective of the heuristic rule proposed in the work was to minimize makespan. Knopf (1985) worked on using branch and bound technique to determine makespan of the given production sequence with intermediate storage in multiproduct batch process.

Wiede et al. (1987), and, Wiede and Reklaitis (1987) proposed scheduling serial multiproduct batch processes with mixed intermediate storage (MIS). They focused on UIS, FIS, NIS and ZW between processing units. The objective function is to find optimum production sequence which offered minimum makespan. Karimi and Ku (1988) proposed a modified heuristic technique to generate the initial sequence for multiproduct batch processes considering transfer and setup time in addition to processing times.

The application of search methods in batch scheduling with parallel units has also been explored by many past researchers. Musier and Evans (1989) observed the importance of product sequencing in production scheduling of industrial batch processes. The heuristics proposed in the solution method is also able to propose optimal or near optimal solutions for batch processes with parallel units. Lee and Lee (1996) developed a method of combining heuristics with NLP formulation to address the preliminary synthesis of multiproduct batch plant. The preliminary synthesis addressed the issues of equipment volumes and mode of operations of non identical parallel units. Patel et al. (1991) considered the design of multiproduct batch plants with different operating modes for non identical parallel units in stage. They combined heuristics with simulated annealing for handling intermediate storage during the design calculation. He and Hui (2006) proposed a rule evolutionary approach for single stage multiproduct batch process with parallel units. The approach is to combine genetic algorithm and tabu search with suitable heuristic rules to obtain near optimal solutions for large size problems. The overall objective is to find the sequence of production that offered minimum makespan.

Das et al. (1990) applied four different simulated annealing algorithms in scheduling multiproduct batch plants under the assumption of permutation schedule with the objective of finding minimum makespan for ZW, NIS, UIS, FIS and MIS. Ku and Karimi (1991b) presented evaluations of a potential simulated annealing method for solving multi-product batch process scheduling problems to minimize the makespan of a serial flow-shop with unlimited intermediate storage. Ryu et al. (2001) used simulated annealing method for optimal scheduling of multiproduct batch process to handle flexible due date and flexible customer requirements. They showed that more than one sequence could meet the due date criteria with minimum earliness and tardiness penalty. The production sequence with minimum makespan was taken as the optimal solution.

Very few researchers have worked on the application of genetic algorithm for multiproduct batch process scheduling. Pozivil and Zdansky (2001) applied the genetic algorithm to find the sequence of batches that minimized the makespan in serial multiproduct batch processes. Caraffa et al. (2001) considered GA for the flowshop problem where no intermediate storage was considered.

Lee et al. (2002) developed a novel list based on threshold accepting (LBTA) algorithm to solve zero wait scheduling problems in multiproduct batch plants. The objective of LBTA was to find the minimum makespan. The LBTA was claimed to give optimal solution for small to moderate size ZW scheduling problems within a very short time.

2.4.2.2 Multipurpose Batch Process

The application of heuristics in the scheduling of multipurpose batch process has also been suggested in the published literature. Janicke et al. (1984) described a simple heuristics to solve scheduling problems in multipurpose batch plants. The heuristics were based on exact algorithm which examined whether or not a batch with a given starting time could be scheduled. Janicke (1987) used a graph based algorithm to address the sharing of equipment in multipurpose batch plants. They determined the number of units, their sizes and the time taken by each unit to complete the required task to meet the production requirement.

The applications of simulated annealing and branch and bound technique for the determination of optimum solution in the scheduling of multipurpose batch plants were suggested by Sanmarti et al. (1998) for NIS and UIS. Azzaro-Pantel et al. (1998) solved the multipurpose batch scheduling problem with a two-stage methodology that involved coupling discrete event simulation with a generic algorithm (GA). The application of GA was also explored in the design of multipurpose batch plants (Bernal-Haro et al. 1998).

Graells et al. (1995; 1996) introduced a comprehensive method for scheduling multi-purpose batch-chemical processes while considering intermediate storage and in-phase and out-of-phase operation modes. Romero et al. (2004) proposed a graph theoretical approach for the optimal scheduling of multipurpose batch plants considering intermediate storage as shared storage i.e. common intermediate storage (CIS), to achieve maximum plant flexibility. They applied branch and bound search algorithm to check the accuracy of their proposed algorithms.

2.5 Summary

The literature survey presented here is just an overview of applying different techniques for solving batch scheduling problems and covers more on the subject of multiproduct batch process. For detail knowledge in the area of batch scheduling and design, the contributions from other authors should also be referred especially on the subject of multipurpose batch process (Castro et al. 2001; 2005; Giannelos and Georgiadis, 2002a,b; Zhu and Majozi, 2001; Majozi and Zhu 2001; Maravelias and Grossmann, 2003a,b). The comprehensive reviews, presented by Rippin (1983a,b), Ku et al. (1987), Mendez et al. (2006) and Barbosa-P′ovoa (2007), cover a wide range of scheduling aspects for batch processes. Although, the available mathematical approaches have been successfully applied in many batch process industries, there is still room for exploring alternate approaches which can equally solve the batch scheduling problem but with reduced computational and mathematical complexity.

As presented and observed from the literature review, the makespan algorithms for solving multiproduct batch process are the basis of any scheduling technique. Also, the optimal value of the process makespan depends on the ways in which the intermediate products are handled between the process stages. However, most of the available methods formulate the batch scheduling problem using MILP and MINLP techniques. So far, most of the scheduling methods apply makespan algorithms developed by Ku and Karimi (1988), Rajagopalan and Karimi (1989), and, Ku and Karimi (1990). Later, these algorithms were modified by Moon et al. (1996), Kim et al. (1996) and Jung et al. (1996) to incorporate some additional variables for considering transfer and setup time in makespan calculation. Lee et al. (2002) proposed a new makespan algorithm based on meta-heuristic approach (LBTA) and applied on the batch scheduling problem taken from Kim et al. (1996).

Generally it is observed that different mathematical methods could not always produce the same optimal sequence for same batch process recipe though makespan calculated is same (Pitty and Karimi, 2008). Further observation can be made from Kim et al. 1996 and Lee et al. 2002. For the same batch process recipe, both of the above papers showed different optimal sequences for ZW and UIS though the makespan calculated is same. Interestingly, again, for ZW and for the same batch process recipe, the optimal sequence as well as makespan is different in Jung et al. 1994 from Kim et al. 1996 and Lee et al. 2002. In actual fact, there are a total of 10 production sequences producing the same makespan as reported in Jung et al. 1994 for optimal sequence but only one was reported. This can be validated by plotting the respective Gantt charts for each sequence. This shows that mathematical methods produce only one optimal solution though more than one solution with same makespan exists. Sometimes, the process planner may be interested in other possible solutions that resulted in the same makespan. For example, if any specific product can not be produced before or after any other specific product. So other possible solution may help in deciding the best one. For this purpose, total enumeration is preferred that searches through all possible solutions before the optimal solution is determined. However, the CPU time increases exponentially with problem size. More products mean more possible solutions to be evaluated before getting the optimal solution (Pitty and Karimi, 2008).

The proposed method in the present work aimed at providing alternate procedure and new makespan algorithms using simple mathematics. In addition, a set of heuristic guidelines have been proposed to reduce the optimal solution search space. The reduction in search space has been made possible using heuristic guidelines in combination with partial enumeration. Contrary to complete enumeration, partial enumeration does not require to search all the possible solutions for the determination of the optimal solution. As a result, the CPU time would also reduce significantly compared with total enumeration.

CHAPTER 3

THEORY OF BATCH PROCESS

3.1 Batch process

Batch process is a process used to produce single or multiple products in batches. Batch processing often require multiple operations, such as mixing, blending, reaction, separation and others. This type of processing involves a number of units or stages arranged in series. An example of a batch process with four stages is shown in Figure 3.1 for the production of C from raw materials A and B. The reaction between raw materials A and B is carried out in a reactor to produce C. Next the mixture from the reactor is mixed with solvent D in a mixer for the purpose of extracting the unreacted raw materials A and B from product C. Product C then formed a solid which is separated from the liquid containing A, B and D in a centrifuge. Finally, a tray drier is used to dry the solid product C (Biegler et al. 1997).

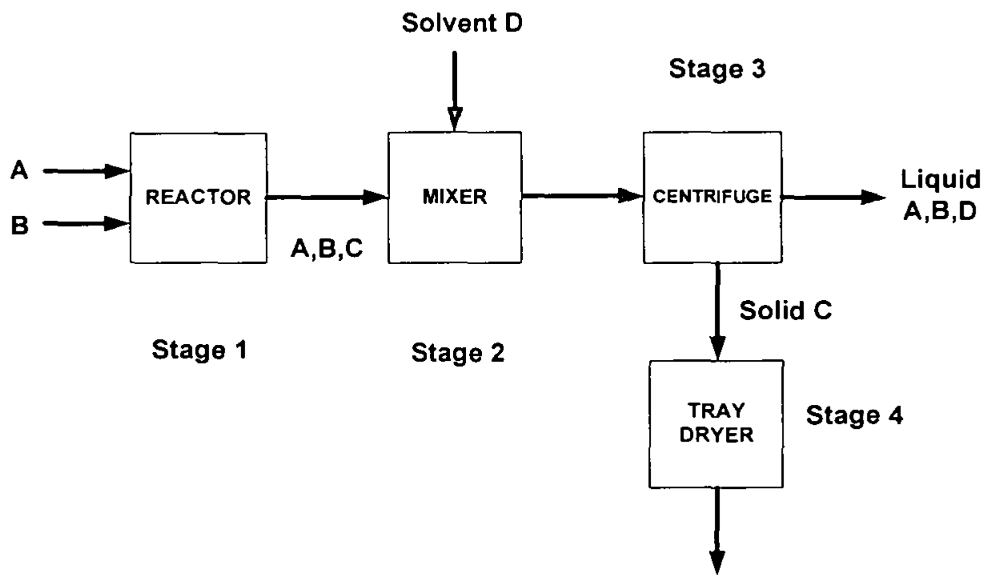


Figure 3.1: An example of a batch process (Biegler et al. 1997)

From the example, it can be easily observed that the processing time for every stage involved in the batch process depends on the type of unit operation and the processing nature required. The sequence of processing time for each stage can be easily represented on a Gantt chart.

3.2 Gantt chart

A Gantt chart is a horizontal bar chart developed by Henry Gantt (Lewis, 1991). The Gantt chart is an effective tool for planning and scheduling operations involving interrelationships between many activities. An initial step in developing a Gantt chart is to specify the set of tasks or activities that make up the process. The amount of time required for each activity is represented as a horizontal bar on the chart. Horizontal bars of varying lengths represent the sequences, timing, and time span for each task. The Gantt chart is easy to construct and understand, even though they may contain great amount of information at times.

In a chemical batch process, Gantt charts are mostly used to work out the completion time i.e. makespan of the batch process. Calculation of makespan requires data particularly on the number of products to be produced and the corresponding batch process recipes i.e. the number of process stages involved and the processing time for each stage. The processing time for every stage is represented on a Gantt chart using the horizontal bars as shown in Figure 3.2 for the same batch process example stated earlier.

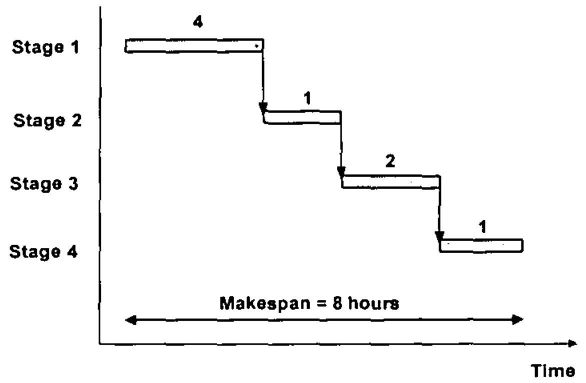


Figure 3.2: Gantt chart (Biegler et al. 1997)

3.3 Modes of batch operation

A batch process can be operated either on a non-overlapping or overlapping mode. In the case of non-overlapping mode, processing of the next batch of products will not commence until the processing of the preceding batch is completed fully. While in overlapping mode, processing of the next batch will begin as soon as the first stage becomes available to process the next batch. The purpose of overlapping the batch process is to reduce the idle time of each process stage. This can significantly reduce the makespan thus increasing plant productivity and efficiency (Biegler et al. 1997). Figure 3.3 shows the two modes of operations using Gantt chart for the same batch process example stated earlier.

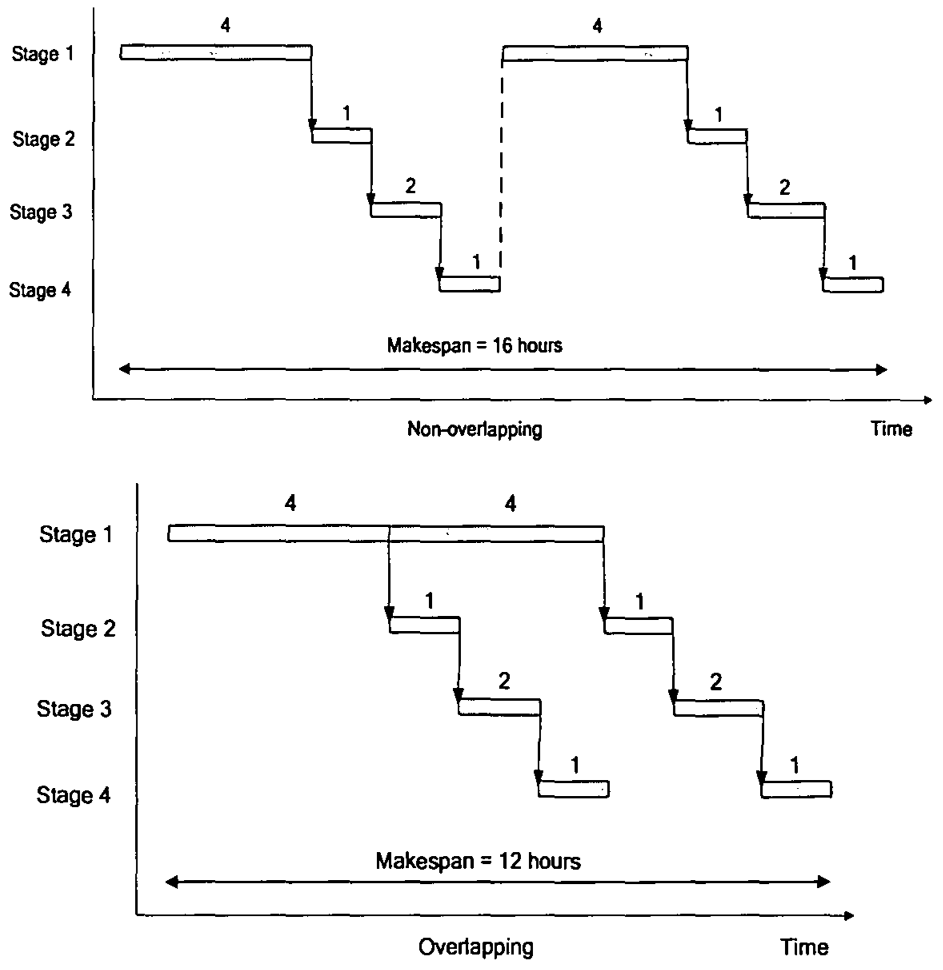


Figure 3.3: Modes of batch operation (Biegler et al. 1997)

3.4 Types of Batch Processes

Batch processes are generally categorized as single product and multiple products. Single product batch process refers to the production of only one type of product in repetition while multiple products batch process produces multiple products using the same batch plant facility. Multiple products batch processes could be further classified as Multiproduct (Flowshop) or Multipurpose (Jobshop) (Biegler et al. 1997).

In multiproduct batch process, all products follow the same sequence of processing and this is illustrated in Figure 3.4 (a). In the case of multipurpose batch process, production of the various products does not have to follow the same processing sequence and also not necessarily utilizing all the process stages, as illustrated in Figure 3.4 (b).

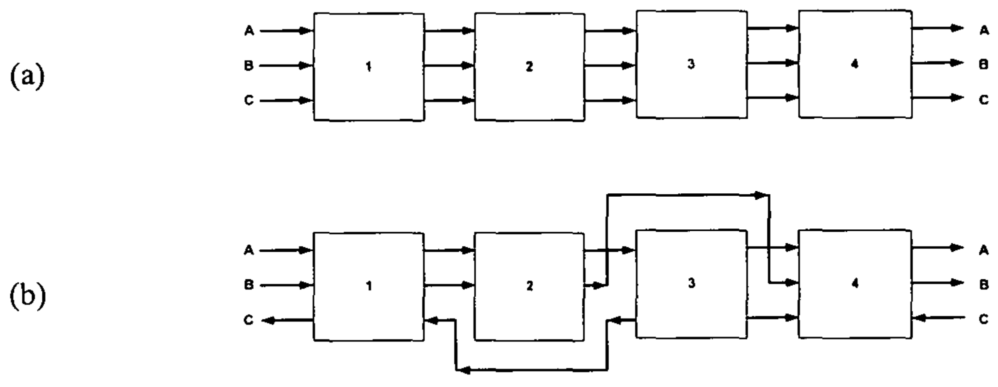
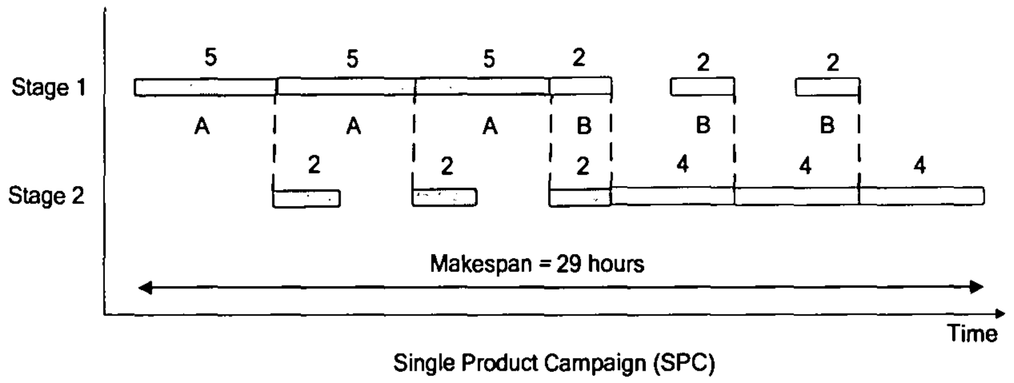


Figure 3.4: Types of batch processes (a) Multiproduct (b) Multipurpose (Biegler et al. 1997)

3.5 Types of production campaigns

A multiproduct batch process can be of single product campaign (SPC) or mixed product campaign (MPC). In single product campaign, all batches of a selected product are produced before the production of another product begins. While in mixed production campaign, various batches of different products can be produced according to some selected sequence (Biegler et al. 1997). The Gantt chart in Figure 3.5 shows the makespan calculation for both campaigns. The advantages of MPC over SPC could be observed in the reduced idle time between stages. However, the situation may change occasionally depending on products recipes (Birewar and Grossmann, 1989).



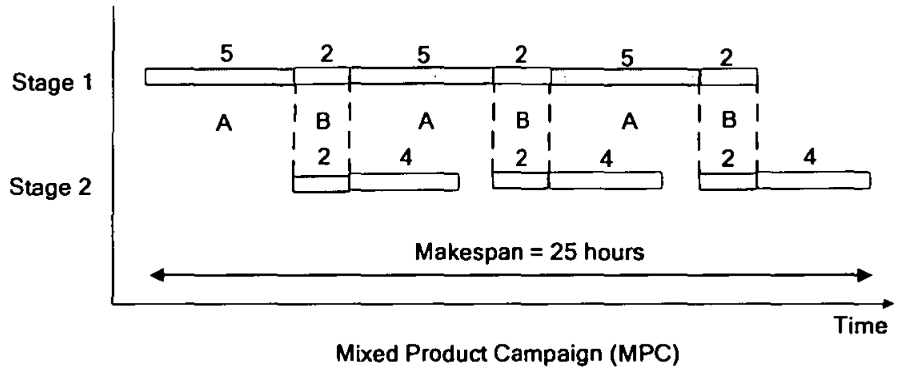


Figure 3.5: Types of production campaigns (Biegler et al. 1997)

3.6 Transfer policies for product intermediates

The transfer of product intermediates from one stage to another in a batch process follows a number of transfer policies as categorized by past researchers. The choice of transfer policy in a batch process depends on two important factors namely the physical and chemical nature of the product being produced and the time frame within which the production must be completed to meet the market demand. The generally adopted transfer polices are discussed below.

3.6.1 ZW transfer policy

In ZW transfer policy, the product intermediate is transferred immediately to the next stage upon completing its respective process due to its nature that requires immediate transfer (Biegler et al. 1997 ; Ryu and Pistikopoulos 2007). Such requirement could lead to a situation where the production of the next batch could not be started immediately upon the availability of the first process stage. This is due to the timing adjustment required to ensure the zero wait transfer policy is observed in all stages and for all products. Figure 3.6 shows the timing adjustment required for a batch process in order to fulfill the zero wait transfer policy. Note that there are a number of discontinuities in some of the production paths such as the delay in starting the process for product B after the product A intermediate has been transferred out from the first stage. This discontinuity refers to the time during which the stage remains idle. The ZW transfer policy often results in the longest completion time or makespan compared to other transfer policies. For instance, in Figure 3.6, it is observed that the processing of product B in stage 3 would only start when the processing of product A has been completed. As a result, the timing for processing of product B in stage 1 and stage 2 has to be delayed in order to meet the requirement at stage 3 in accordance to the rules of zero wait transfer policy.

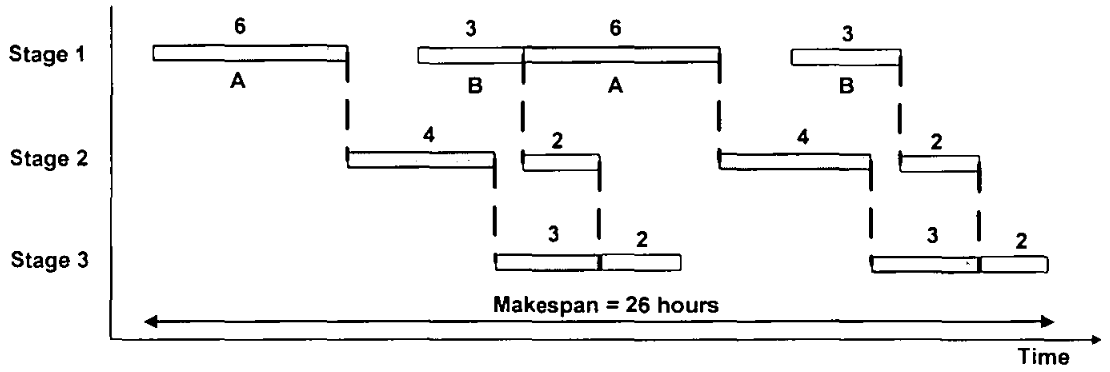


Figure 3.6: Gantt chart for ZW transfer policy (Biegler et al. 1997)

3.6.2 NIS/UW transfer policy

In NIS/UW transfer policy, the nature of the intermediates is such that they could stay in their current stage until the availability of the next processing stage (Ku and Karimi 1992; Biegler et al. 1997). In this way, the completion time of a batch process can be reduced compared to the ZW transfer policy as illustrated in Figure 3.7. The reason is because the processing of the next product can commence immediately upon the availability of the required processing stage. Figure 3.7 shows that the processing of product B in stage 1 could start immediately after the processing of product A and does not have to depend on the timing availability of stage 2 since the product intermediate could reside temporarily in stage 1 if required.

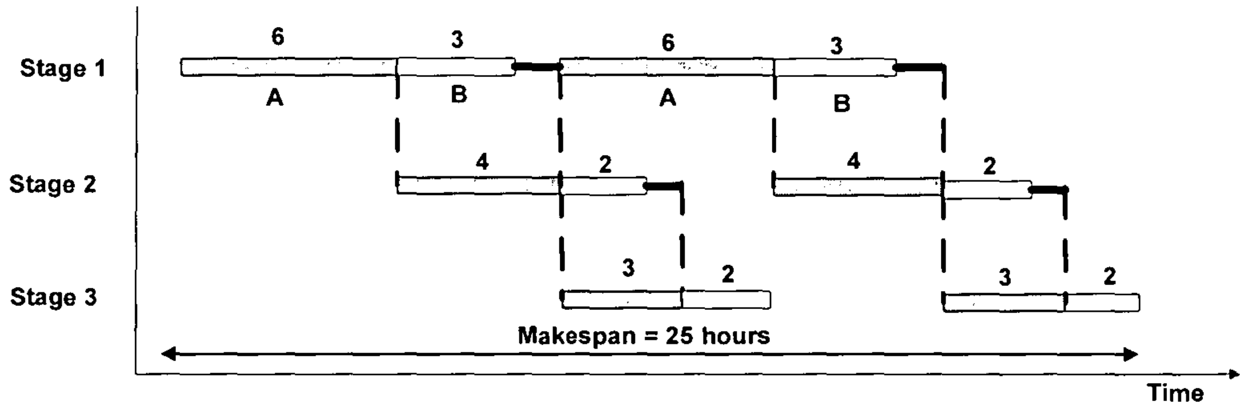


Figure 3.7: Gantt chart for NIS/UW transfer policy (Biegler et al. 1997)

3.6.3 UIS/UW transfer policy

In UIS/UW transfer policy, intermediate storage tanks are used to store the product intermediates temporarily until the availability of the next process stage. This situation is adopted when the product intermediate is not allowed to reside temporarily in the same process' stage due to either process makespan restriction or product intermediate undergoing further reaction if remains within the process stage (Biegler et al. 1997).

Similar to NIS/UW, the processing on every stage in UIS/UW does not depend on the timing availability of the next stage i.e. the production of next product in any stage can be started soon after the production of earlier product has been completed. However, in the case of unavailability of the next stage, the product intermediate does not wait inside the same stage as it does in NIS/UW rather it is transferred to the temporary storage as shown in Figure 3.8.

Due to the unlimited number of storages made available, there is no restriction at all on the temporary storage of product intermediates as shown in Figure 3.8 (Kim et al. 1996). For instance, the temporary storages available in Figure 3.8 are for storing the product B intermediate after stage 1 and stage 2. Due to the physical and chemical nature of the product intermediates, the residence time in a temporary storage must be monitored carefully to meet the quality standards of the final product (Ha et al. 2000).

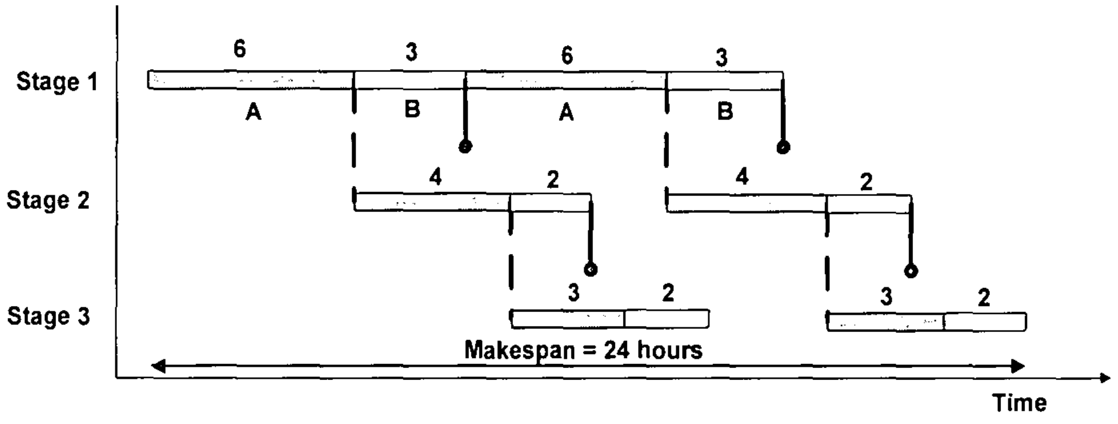


Figure 3.8: Gantt chart for UIS/UW transfer policy (Biegler et al. 1997)

3.6.4 FIS/UW transfer policy

In the case of FIS/UW, the process features are the same as the UIS/UW except that the number of storages is limited. The storage system in FIS transfer policy results in better economics compared to the UIS/UW as it tend to reduce the capital cost while optimizing the storage utilization (Kim et al.1996).

Kim et al. (1996) suggests the application of FIS/UW by combining the process features of FIS/UW and NIS/UW. The combination works in such a way that whenever a storage is available, the product intermediate is transferred into it else the product intermediate will reside temporarily in its current process stage (NIS/UW) until the availability of the temporary storage. For example, in Figure 3.9, a temporary storage is used to store the product B intermediate after its processing in stage 1 has been completed. This is due to the unavailability of stage 2 at that point of time which is still processing the product intermediate of A. The need of temporary storage again arises after producing the second batch of product A in stage 1. However, at that point in time, the temporary storage is storing product intermediate of the first batch of product B. Therefore, the product intermediate of the second batch of product A must be held inside stage 1 i.e. NIS/UW is observed until the availability of temporary storage as shown in Figure 3.9.

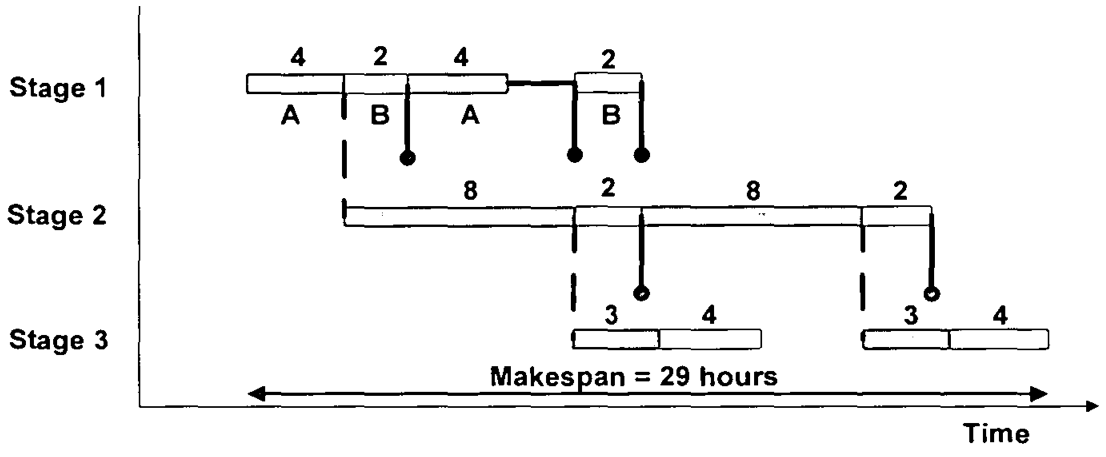


Figure 3.9: Gantt chart for FIS/UW transfer policy

3.6.5 MIS/UW Transfer Policy

In MIS/UW, any of the previously discussed transfer policies could be applied in combination with other in between any of the two stages in a multiproduct batch process. For instance, in a 3 stage batch process shown in Figure 3.10, nature of product B intermediates is such that they can stay inside the stage 1 in case the next stage is not available. However, the process conditions do not allow the product B intermediate to stay in stage 2 in case stage 3 is not available. Hence, there is a need of temporary storage tank just after stage 2 as soon as processing of product B intermediates completes in stage 2. In other words, the intermediates follow NIS/UW in between stage 1 and 2 while follow UIS/UW in between stages 2 and 3 as shown below in Figure 3.10.

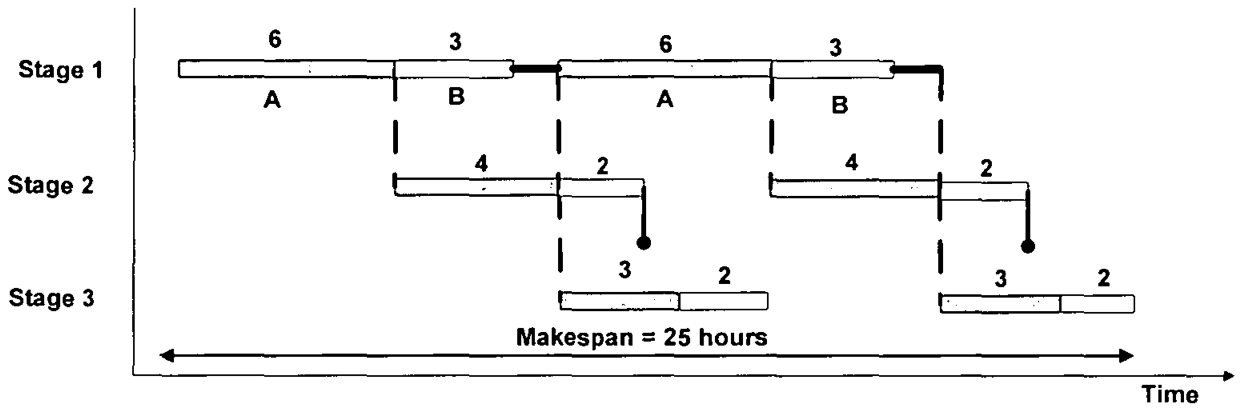


Figure 3.10: Gantt chart for MIS/UW transfer policy

3.7 Batch process scheduling

The selection of the right transfer policy plays a significant role when producing various products in a multiproduct batch process. The production of various products may vary in order to meet the frequent changes of market demand. This necessitates an efficient production schedule that controls the sequence and timing of different operations to produce the different products. It has forced industry to make effective utilization of available resources using proper scheduling systems.

A typical batch process scheduling problem depends on the following specifications:

- Transfer polices for product intermediates between processing stages.
- Processing order of various products.
- Transfer and setup time between different processing stages.

The quality of a batch process schedule can be measured by one or a combination of the following criteria:

- Meet production requirement within the specified time period
- Meet customer's requirements on quantity and quality standards
- Minimum completion time i.e. makespan of the batch process
- Minimum installation, production, handling and inventory costs
- Maximum utilization of manpower and process equipments
- Safe and environment friendly operation

3.7.1 Batch process scheduling benefits

The benefits of batch process scheduling may vary from product to product. However, some of the benefits are common among the different products and these were explained by Morrison (1996) as follows:

- The number of equipments involved in the production process can be optimized to minimize the cost and labor requirements. This can be achieved by adopting proper sequence of the products being produced.

- The excess of inventory could result in extra costs incurred in maintaining the quality of the stored products. Effective scheduling can help in managing the inventory level of products according to the raw material supply and to meet the sudden changes in the product demand.

- The production time should be able to meet the due date set by the customers. The effective scheduling can decide the order of the products that can reduce the overall production time.

- The most important benefit of scheduling is its flexibility to manage the unforeseen events such as equipment breakdown, rush orders, order changes and raw material availability.

3.7.2 Batch process scheduling techniques

The benefits of batch process scheduling can be achieved by adopting effective scheduling techniques. One of the important aspects in batch process scheduling is to minimize the total completion time i.e. makespan of a batch process. The production capacity of a batch plant can be maximized if all the production tasks would be completed within a minimum span of time. A number of scheduling techniques have been developed to meet the minimum makespan. These techniques mainly depend on the type of batch processes such as single product, multiproduct and multipurpose batch process.

In single product batch process, only one type of product is produced. The task of achieving minimum makespan for single product batch process is usually achieved by installing additional process unit parallel to the existing one. The additional unit is usually installed in parallel to the existing unit within the stage that has the longest processing time. This technique is only suitable and recommended for single product batch process. This is in view of the fact that the same process stage requires the longest processing time for various batches of the single product. However, the technique is not recommended for multiproduct or multipurpose batch processes in view of the changes in the process stage that requires the longest processing time for various products (Ryu et al. 1997).

The task of achieving minimum makespan in multiproduct and multipurpose batch processes could be achieved by proper product sequencing. The product sequencing is the order in which the products are produced using the same batch facility. The makespan of the batch process normally varies with the changes in the production sequence. The optimum production sequence could be taken as the one that offers the least makespan.

The product sequencing works on its best by making efficient use of the idle time that exist in between process stages. The idle time is the time during which a process unit remains idle and causes delay in the processing of the next product. The period of idle time varies with products recipe (Kim et. al 1996, Moon et al. 1996).

Contrary to multiproduct and multipurpose batch processes, product sequencing is not applicable for single product batch process. This is because in single product batch process, multiple batches of only one product are produced using the same batch facility.

Product Sequencing

The optimal batch process schedule is often based on process planner's choice for the production sequence offering minimum makespan. The possible number of production sequences could be determined using a simple permutation rule shown below:

$P(n) = n!$ where $P(n)$ = number of possible production sequences

n = number of products

For example, the number of possible production sequences for three products namely A, B and C is $P(3) = 3! = 6$ i.e. ABC, ACB, BAC, BCA, CAB and CBA.

The determination of makespan for any production sequence can be done using the Gantt chart method described earlier. However, this becomes tedious and not recommended for large size problems. Due to this limitation, researchers have proposed many computational techniques involving mathematical algorithms which determine the production sequence that offers the least makespan.

Most of these mathematical algorithms apply mathematical equations based on linear and non linear programming. For speedy computations, these mathematical equations are solved using computational software specially designed for the purpose. An overview of some of the popular methods for batch process scheduling is given below.

3.7.2.1 Mathematical Methods

One method of solving batch scheduling problem is using mathematical programs. A mathematical program is a way to define scheduling problem using variables, constraints and objective functions. Variables can be continuous or discrete according to the problem being addressed. These variables are used to define task assignment to various units, time allocation, batch sizes and allocation of storages between process stages. The mathematical programs can be extended using additional constraints and variables for more detailed process description. These are further classified as mixed integer linear program (MILP) and mixed integer non linear program (MINLP).

1. Mixed Integer Linear Program

A mixed integer linear program (MILP) is a linear program (LP) in which one or more than one variable have to be integers. The most commonly used subset in MILP is binary in which the integer variables can be either 1 or 0 meaning that something is either present or not present (Edgar et al. 2001). As an example, in a batch scheduling problem, the binary variable controls the presence of any product at a specific position in a particular production sequence (Kim et al. 1996).

2. Mixed Integer Non Linear Program

A mixed integer non linear program (MINLP) is used to solve nonlinear optimization problems with both continuous and discrete variables (Edgar et al. 2001). As such the MINLP is more complex compared to MILP. In batch scheduling problem, the non linearity is usually attributed to the addition of sequence dependent setup time for all process stages and thus, has to be formulated as MINLP (Kim et al. 1996).

3. Branch and Bound

The Branch and Bound (B&B) is a general search method which is used to solve many batch scheduling problems which are usually represented by MILP and MINLP and mostly subjected to some equality or non equality constraints. In B&B procedure, these constraints are used to assign lower and upper bounds to set a criterion to find the optimal solution in a feasible solution region.

The B&B method starts by considering the original problem with its complete feasible solution region. The original problem is then analyzed according to the lower and upper bounds. The procedure ends only if the optimal solution is found. Otherwise, the feasible region is divided into two or more regions (subproblems) represented as nodes. The algorithm is then applied to the subproblems to determine the optimal solution. However, it is not necessary that the solution obtained is globally optimal. The node can be removed from further analysis if it does not meet the criteria to find the optimal solution. The search procedure continues until all the nodes have been analyzed for the global optimal solution (Edgar et al. 2001).

3.7.2.2 Heuristics and Metaheuristics

A heuristic technique is a method which generates optimal or near optimal solutions within a reasonable time frame. Despite the fact that it does not always guarantee optimal solution, the present heuristics methods could produce reasonably optimal solutions for large size problems within shorter time period compared to the mathematical programming. In addition, the method is also known to be more stable.

The basis of many heuristic methods is the neighbourhood search. It is a simple iterative method for finding good solutions. The procedure starts from assigning the values to the variables. The search process continues until no further improvement is possible. In recent years, many improvements in the local based search methods have been made to overcome the local minimum phenomena and to look for other possible solution in the effort to find the global optimum. The heuristics techniques with such improvements are now known as metaheuristic techniques.

1. Simulated Annealing

Simulated Annealing (SA) is a class of metaheuristics algorithm for finding the global optimum solution for a given objective function in a large search space. SA was developed based on the analogy from the annealing process of metals in metallurgy, a technique involving heating and controlled cooling of a material to produce perfectly structured crystals. The heat causes the atoms to be dislocated from their initial positions with low internal energy and move randomly in states of higher energy. The slow cooling provides better chances for them to find configurations with lower internal energy than the initial one (Edgar et al. 2001).

In a batch process scheduling problem, the objective function is to find the production sequence with minimum makespan. The simulated annealing starts with an initial solution i.e. a production sequence with some makespan value, followed by comparison with the makespan of the second possible production sequence. The comparison process continues till the search space having all the possible solutions is analyzed.

2. Tabu Search

Tabu Search (TS) belongs to the class of local search techniques and widely used in solving many planning and scheduling problems. Most heuristic methods fail to locate global optima because they usually get trapped in local optima i.e. the search does not continue for other near optimal solution as soon as the first solution was found. Tabu search overcomes this limitation by maintaining a tabu list containing the solutions which have already been searched for optimal solution. The term tabu means "forbidden". This procedure keeps the tabu search away from revisiting the previously visited solutions.

The search continues until there is no more optimal solution within the solution search space (Tra, 2000, Edgar et al. 2001).

3. Genetic Algorithm

The concept of Genetic Algorithms (GA) is taken from the evolution of new living organisms. The new living organisms are generated by crossover and mutation of the chromosomes from the parents during reproduction process. As a result of reproduction, an initial population is created. Before beginning a new iteration, the population is modified by replacing one or more individuals with new generations. The new generations are usually created using two methods i.e. either by combining two individuals known as crossover or by changing an individual known as mutation. In each generation, the individuals meet the fitness level reproduce while the others do not take part in the next reproduction. (Edgar et al. 2001).

In batch scheduling, the production sequences or schedules could be considered as individuals or members of a population. The individuals are sometimes referred to as chromosomes which carry the information on sequence of products on all stages. The fitness or survival of each individual in the next generation is measured using the value of the objective function i.e. the computation of the makespan in most cases. During iteration, new generations i.e. production sequences are created. The new generations are created by crossover or mutation of the previous generations. The mutation process on the parent chromosomes is similar to the change of positions of two products with each other in the corresponding production sequence. The population size i.e. number of products in a particular production sequence usually remains constant in each generation (Pozivil and Zdansky, 2001).

3.8 Summary

The transfer of product intermediates from one stage to the next in a batch process is governed using various transfer policies. The selection of a transfer policy for a particular batch process depends on the production targets and the constraints set by the material processed in order to achieve customer specification. Batch scheduling problems under different transfer policies can be solved using various available methods. The mathematical programming methods such as MILP and MINLP are the most popular methods as they could incorporate most of the batch process requirement using various variables and constraints. However, the computational time increases as the problem size grows. As such, researchers have introduced other methods known as heuristics and metaheuristics. These methods are faster and more stable in their computation but seldom guarantee optimal solution. The present work is about proposing an alternative method which offers reduced complexity in solving the batch scheduling for any batch process recipe operated under different transfer policies. Other features of the proposed method include development of some heuristic guidelines to reduce the computational time in searching for the optimal solution.

CHAPTER 4

MAKESPAN DETERMINATION USING MATRIX REPRESENTATION

4.1 Gantt Chart Method

In chemical batch process scheduling, Gantt charts are used to represent the processing times on various stages in a batch process which could help in determining the completion time i.e., makespan of the process. Calculation of makespan requires data particularly on the number of products to be produced and the corresponding batch process recipes, containing the number of process stages and the processing time for each stage. Based on a selected production sequence, the data could be used to draw a Gantt chart before the makespan could be determined.

4.2 The Proposed Matrix Representation Method

While the Gantt chart method looks relatively simple and capable of determining makespan of a batch process, it is actually tedious to perform particularly for large number of products. On the other hand, most previous research conducted on batch scheduling focused on the use of complex mathematical methods namely the mixed integer linear or non-linear programming. Although these methods are highly efficient in execution, formulation of the batch scheduling problem could pose a problem to most planners as it requires considerable understanding in the use of high level mathematics. The proposed method offers an alternative to batch scheduling efficiently by avoiding the use of MILP and MINLP. Process planners are only required to input the data of batch process problem and the required information regarding makespan could be determined.

The development of the proposed procedure resulted from the observation made on the paths used for calculating makespan using Gantt chart method. Suppose there are three products to be produced namely A, B and C in the sequence of producing A first, followed by B and lastly C using three stages S_1, S_2 and S_3 as shown by the Gantt chart in Figure 4.1 below.

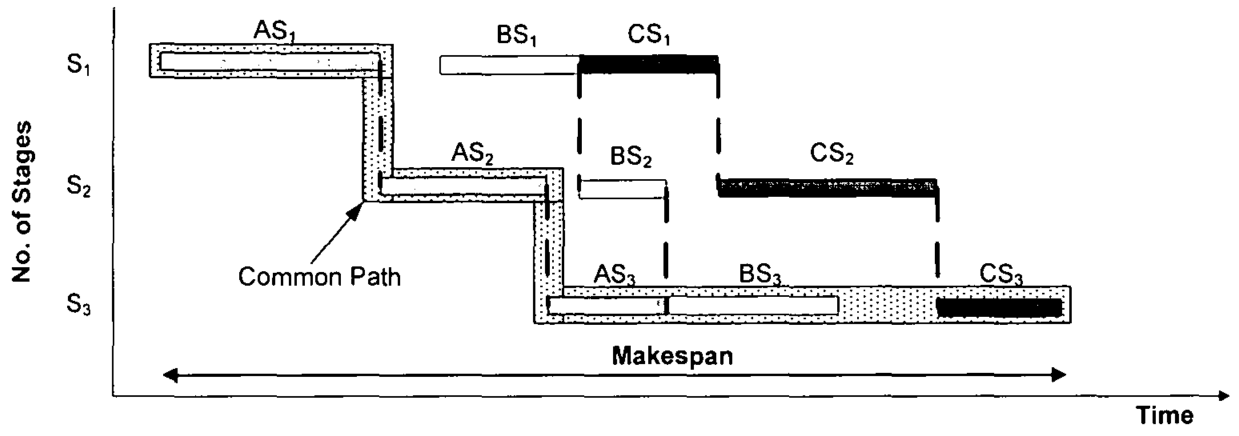


Figure 4.1: Gantt chart of three products in three stages

The makespan for the batch process could be calculated based on several identified paths connecting the starting point to the end point as shown in Figure 4.1. Each path contains the respective processing times of the various products on the different process stages. Summing all the processing times and the idle times that exist within the path will produce the makespan. Some of the possible identified paths are listed below:

1. AS_1, AS_2,AS_3,BS_3,CS_3
2. AS_1, AS_2, BS_2, BS_3,CS_3
3. AS_1,AS_2,BS_2,CS_2,CS_3
4. AS_1,BS_1,CS_1,CS_2,CS_3
5. AS_1, BS_1,BS_2,BS_3,CS_3
6. AS_1, BS_1,BS_2,CS_2,CS_3

It is observed from Figure 4.1 that regardless of the path selected, the result of the makespan calculation for a selected production sequence derived from a given batch process recipe would remain the same. This is because the length of all the paths from beginning (AS_1) to the end (CS_3) of the process in a particular production sequence remains the same. The length of any path is calculated by adding up the processing times of each stage and the idle times present in the path. The idle times are reflected by the discontinuity in the path located in between stages at one or more points within the path. For example, there is a path discontinuity in between AS_1 and BS_1, AS_2 and BS_2, BS_2 and CS_2, and, BS_3 and CS_3. For determination of the makespan using such paths, the calculation of idle time, if exists between process stages, is required.

For the purpose of developing a consistent procedure to calculate the makespan for any batch process recipe, it can be concluded that only one common path is required. In the present work, the first path i.e., AS_1, AS_2, AS_3, BS_3, CS_3, from the aforementioned list of possible identified paths, is selected as the common path and it is shown in Figure 4.1.

The batch process recipe from Figure 4.1 can be represented in the form of a matrix as shown below.

$$\begin{array}{cccc} & 1 & 2 & 3 \\ 1 & AS_1 & AS_2 & AS_3 \\ 2 & BS_1 & BS_2 & BS_3 \\ 3 & CS_1 & CS_2 & CS_3 \end{array}$$

For the purpose of systematic execution of the proposed method, the respective products to be produced are arranged according to the rows represented by the letter "i" and the respective stages are arranged according to the columns represented by the letter "j". Hence, for the case presented, the matrix used would be "M_{ij}" where i=1,2,3 and j=1,2,3.

In this respect the scheduling is based on a sequence where product A is produced first, followed by B, and lastly C.

The next step is to recognize the presence of idle time between stages by introducing variables 'V' in between the rows of the matrix. In the matrix shown, there are possible six variables that can be introduced namely $V_{1,1}$ located in between AS_1 and BS_1, $V_{1,2}$ located in between AS_2 and BS_2 ,$V_{1,3}$ located in between AS_3 and BS_3, $V_{2,1}$ located in between BS_1 and CS_1, $V_{2,2}$ located in between BS_2 and CS_2 and $V_{2,3}$ located in between BS_3 and CS_3.

$$\begin{array}{cccc}
 & 1 & 2 & 3 \\
1 & AS_1 & AS_2 & AS_3 \\
 & V_{1,1} & V_{1,2} & V_{1,3} \\
2 & BS_1 & BS_2 & BS_3 \\
 & V_{2,1} & V_{2,2} & V_{2,3} \\
3 & CS_1 & CS_2 & CS_3
\end{array}$$

Determining the values of these variables is important as it contributes towards the calculation of the makespan based on the selected path as shown in the earlier Gantt chart. The manner, in which these variables are determined, depends on the transfer policies adopted for the batch process operation.

4.2.1 Makespan determination for various intermediate transfer policies

The development of the proposed method using matrix representation is best explained using examples of batch processes. For all the forthcoming examples, the batch process recipes for all the products are limited initially to three process stages. The transfer time of the product intermediates from one stage to the other and the setup time for each stage are assumed negligible at this point.

4.2.1.1 ZW transfer policy

Example 4.1 and 4.2 below demonstrates the application of the Gantt chart method to determine the makespan of the batch process where each product follows the same operation sequence for all the process stages.

Example 4.1

Table 4.1 shows the batch process recipes arranged accordingly for a production sequence producing two products namely A followed by B. Using the Gantt chart as shown in Figure 4.2, it is observed that there are two locations where idle time exist in between the process i.e., at the end of stage 1 and stage 3 (shaded area), while there are none for stage 2. Table 4.2 shows the summary of the calculated idle time obtained from the Gantt chart.

Careful examination on the Gantt chart reveals that the calculation of makespan could be done using four different paths. One path is to take the sum AS_1, AS_2, BS_2 and BS_3 which does not require any calculation of the idle time. However, the other three paths i.e., either taking the sum of AS_1, BS_1,BS_2,BS_3 or the sum of AS_1, BS_1, AS_3, BS_3 or the sum of AS_1, AS_2, AS_3, BS_3, need to account for the idle time calculation prior to determining the makespan. As expected, the makespan calculated by all four paths yielded the same answer i.e., 45 hours.

Table 4.1
Processing time for production sequence AB for example 4.1

Products	Processing time (hour)		
	S_1	S_2	S_3
A	10	20	5
B	8	12	3

Table 4.2
Idle time calculated for production sequence AB for example 4.1

Products	Idle time (hour)		
	S_1	S_2	S_3
A & B	12	0	7

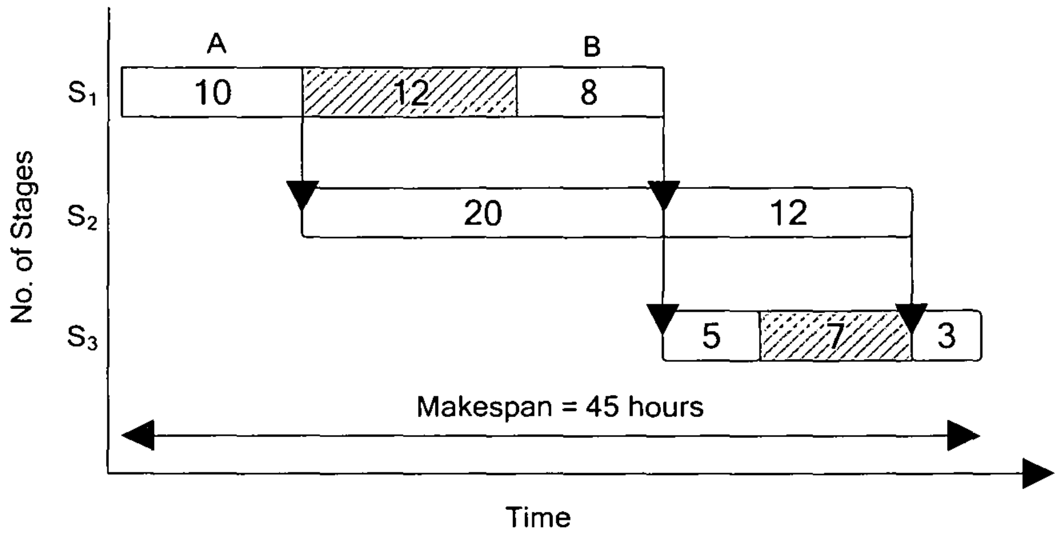


Figure 4.2: Gantt chart for production sequence AB for example 4.1

Example 4.2

In this example, the makespan calculation is performed for batch process producing three products namely A, B and C in the order of product A first, followed by B and lastly C. The batch process recipes based on a 3-stage operation i.e., S_1, S_2 and S_3, are shown in Table 4.3.

From the constructed Gantt chart as shown in Figure 4.3, the idle time locations were found to be at the end of stage 1 and stage 3 for both products. The detail of the result is summarized in Table 4.4. The value of the makespan calculated from the Gantt chart based on the identified paths is 50 hours.

Table 4.3
Processing time for production sequence ABC for example 4.2

Products	Processing time (hour)		
	S_1	S_2	S_3
A	10	20	5
B	8	12	3
C	5	6	2

Table 4.4
Idle time calculated for production sequence ABC for example 4.2

Products	Idle time (hour)		
	S_1	S_2	S_3
A & B	12	0	7
B & C	7	0	3

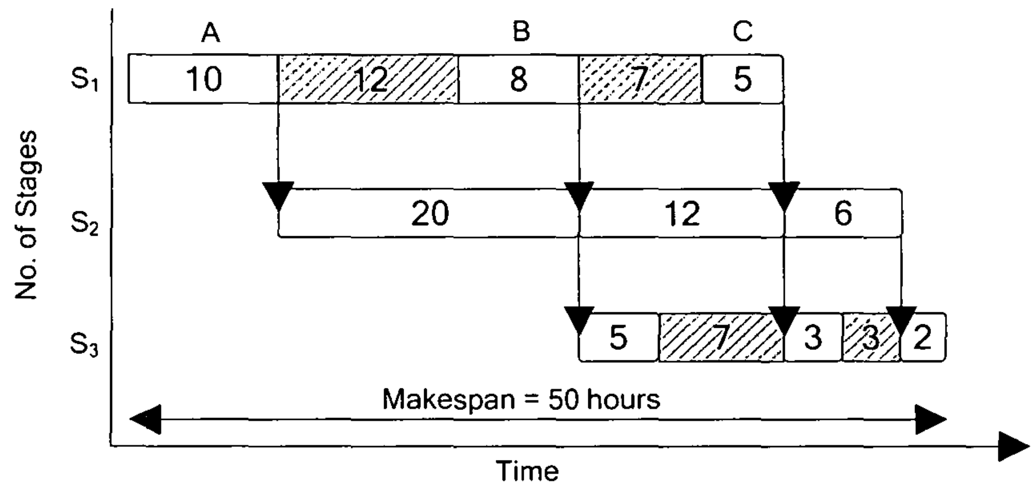


Figure 4.3: Gantt chart for production sequence ABC for example 4.2

Careful examination of Figure 4.3 shows that there are more possible paths to determine the makespan compared to the earlier example which has four. However, the makespan calculation result remains the same irrespective of the path selected.

From the observations made in the two examples, it can be concluded that the makespan, and the idle time period and locations, varies with different batch process recipes. Although the procedure for determining both the makespan and the idle time using Gantt chart method appears to be relatively simple, it is expected to become extremely tedious as the problem size grows.

It can also be concluded that the number of paths to calculate makespan increases with increase in the number of products and also depend on the batch process recipes. But the calculated makespan remains the same from all the possible paths identified. This is in accordance with the observation made earlier that the length of all the paths from beginning to the end points remains the same. This offers flexibility in choosing any of the paths for calculating the makespan. Therefore, as suggested earlier, a common path is selected i.e., AS_1, AS_2, AS_3, BS_3, CS_3, meaning to say that the path which calculates the makespan by taking the sum of the processing times of all stages of first product and last stages of all other products. Of course, the makespan calculation would also include the calculation of idle time, if exist in between process stages for the selected common path. The proposed method is developed on the basis of this observation, to calculate the makespan for "n" number of products to be processed in "m" number of stages.

Matrix Representation

As mentioned earlier, the determination of makespan using the proposed matrix representation can be done by representing the batch process recipe with variables 'V' in the form of a matrix as shown below.

$$\begin{array}{cccc}
 & 1 & 2 & 3 \\
1 & AS_1 & AS_2 & AS_3 \\
 & V_{1,1} & V_{1,2} & V_{1,3} \\
2 & BS_1 & BS_2 & BS_3 \\
 & V_{2,1} & V_{2,2} & V_{2,3} \\
3 & CS_1 & CS_2 & CS_3
\end{array}$$

It is observed from Figure 4.1 that the idle time between stages BS_2 and CS_2 can be determined by taking the difference of processing times for stages CS_1 and BS_2. Similarly, the idle time between stages BS_3 and CS_3 can be determined by taking the difference of processing times for stages CS_2 and BS_3. However, it is obvious from Figure 4.1 that the idle time calculated earlier between BS_2 and CS_2 has to be included for the calculation of the idle time between BS_3 and CS_3. This type of procedure, in which the calculation of idle time for succeeding stage is done using the preceding stage, is termed as forward calculation procedure in this work. This also implies that the calculation of idle time in between the next stages must also incorporate the idle time, if exists, in between the earlier stages.

It is important to note that the procedure for idle time calculation using the forward calculation may not necessarily work for the arrangement of process stages other than observed above for products B and C. For example, the arrangement of stages in between products A and B results in producing the idle time in between AS_1 and BS_1. This situation is different from the case of products B and C observed earlier where there was no idle time in between BS_1 and CS_1. Therefore, for the case of products A and B, the determination of idle time is required between AS_1 and BS_1 before it could be used for the determination of idle time in between the next stages. This is in view of the observation made above i.e., the calculation of idle time in between the next stages depends on the idle time, if exists, in between the earlier stages.

Further it is observed from Figure 4.1 that idle time is also present in between AS_2 and BS_2. Hence, its determination is also required similar to the case for its determination in between AS_1 and BS_1. However, the idle time is none in between AS_3 and BS_3. This implies that for the present case the calculation of idle time for earlier stages could be done starting from the last stage i.e., the idle time between AS_2 and BS_2 can be determined by simply taking the difference of processing times for AS_3 and BS_2 as shown in Figure 4.1. This would be followed by determination of idle time in between AS_1 and BS_1 by taking the difference of processing time for AS_2 and BS_1 while also including the idle time calculated in between AS_2 and BS_2. This type of procedure in which the calculation of idle time for preceding stage is done using the succeeding stage is termed as reverse calculation procedure in the present work.

The difference in the arrangement of stages in between any two consecutive products is due to the transfer policy adopted which is ZW in the present case. It is also noted that there will always be at least one point in between the two consecutive products where the idle time is none as shown in Figure 4.1 for the case of AS_3 and BS_3, and, BS_1 and CS_1. In view of the above, the procedure to calculate the idle time for different arrangements of stages in between any two consecutive products, couples the features of forward and reverse calculation and is shown below.

The idle times represented as variable $V_{1,2}$ and $V_{1,3}$ in the above matrix are calculated by performing the following procedures to the elements of the first two rows located above and below these variables. For example, in Figure 4.3, the variable $V_{1,2}$ can be calculated by subtracting the processing time of product A intermediate in stage S_2 from the processing time of product B in stage S_1, and, the variable $V_{1,3}$ can be calculated by subtracting the processing time of product A intermediate in stage S_3 from the sum of the processing time of product B intermediate in stage S_2 and the variable $V_{1,2}$.

$$V_{1,2} = BS_1 - AS_2$$

$$V_{1,3} = (V_{1,2} + BS_2) - AS_3$$

For the calculation of variable $V_{1,1}$, and to meet the criteria of ZW, a reverse calculation is required to calculate the values of all previous variables 'V' on the basis of the value of the variable $V_{1,3}$. As such, the value of the variable $V_{1,2}$ is recalculated. This is followed by the calculation of variable $V_{1,1}$ which is based on the value of variable $V_{1,2}$ as shown below.

$V_{1,2} = (V_{1,3} - BS_2) + AS_3$

$V_{1,1} = (V_{1,2} - BS_1) + AS_2$

The same procedure is repeated for the calculation of variables $V_{2,2}$ and $V_{2,3}$ between second and third row as shown below:

$V_{2,2} = CS_1 - BS_2$

$V_{2,3} = (V_{2,2} + CS_2) - BS_3$

Based on the value of variable $V_{2,3}$, $V_{2,2}$ is recalculated. This is followed by the calculation of variable $V_{2,1}$ which is based on the value of $V_{2,2}$ as shown by the equations below:

$V_{2,2} = (V_{2,3} - CS_2) + BS_3$

$V_{2,1} = (V_{2,2} - CS_1) + BS_2$

It must also be noted that for a negative value obtained for any variable 'V' above, a zero value will be assigned instead. This actually means that there is no idle time in between the particular stages of reference.

From the calculated values of these variables above, the makespan for the multi product batch process can be calculated using the path selected as shown in example 4.2. The makespan equation is as shown below.

$Makespan\ = AS_1+AS_2+AS_3+V_{1,3}+BS_3+V_{2,3}+CS_3$

The generalized mathematical expression for the matrix representation

The mathematical expressions developed so far to calculate the variables 'V' and the makespan are suited to three product batch processes with three process stages. To apply the same procedure for more than three products and three stages, a certain generalization is required. For this purpose, the variables 'V' in all the above mathematical expressions are rewritten with the subscripts "i" and "j". This represents that the "n" number of products "i" are produced in "m" number of stages "j". The calculation procedure begins with the first two rows of the matrix and then carried forward to the succeeding rows using equations 4.1 and 4.2 respectively.

$$V_{i,2} = M_{i+1,1} - M_{i,2} \qquad i = 1......n\text{-}1 \qquad (4.1)$$

$$V_{i,j+1} = (V_{i,j} + M_{i+1,j}) - M_{i,j+1} \qquad j = 2........m\text{-}1, \;\; i = 1......n\text{-}1 \qquad (4.2)$$

The variables 'V' between the row elements are recalculated based on the calculated value of the last variable using equation (4.3).

$$V_{i,j} = (V_{i,j+1} - M_{i+1,j}) + M_{i,j+1} \qquad j = m\text{-}1........1, \quad i = 1......n\text{-}1 \qquad (4.3)$$

For any negative value of V, zero value will be assigned instead. Using the calculated values obtained for all the variables, the makespan is calculated using equation (4.4).

$$Makespan = \sum_{j=1}^{m} M_{1,j} + \sum_{i=2}^{n} M_{i,m} + \sum_{i=1}^{n-1} V_{i,m} \qquad n \geq 2 \;, m \geq 2 \qquad (4.4)$$

The following example 4.3 illustrates the application of the proposed method to determine the makespan using batch process recipes taken from Ryu and Pistikopoulos (2007) for which he has applied MILP method from Jung et al. (1994). The process recipes according to a selected production sequence are shown in the matrix below.

Example 4.3

$$\begin{array}{cccc} & 1 & 2 & 3 \\ 1 & 10 & 20 & 5 \\ & V_{1,1} & V_{1,2} & V_{1,3} \\ 2 & 15 & 8 & 12 \\ & V_{2,1} & V_{2,2} & V_{2,3} \\ 3 & 20 & 7 & 9 \end{array}$$

The variables $V_{1,2}$ and $V_{2,2}$ are calculated using equation (4.1), and, $V_{1,3}$ and $V_{2,3}$ are calculated using equation (4.2) as follows:

$$V_{1,2} = M_{2,1} - M_{1,2} = 15 - 20 = -5 = 0$$

$$V_{2,2} = M_{3,1} - M_{2,2} = 20 - 8 = 12$$

$$V_{1,3} = (V_{1,2} + M_{2,2}) - M_{1,3} = (0 + 8) - 5 = 3$$

$$V_{2,3} = (V_{2,2} + M_{3,2}) - M_{2,3} = (12 + 7) - 12 = 7$$

On the basis of the calculated values of $V_{1,3}$ and $V_{2,3}$, the values of $V_{1,2}$ and $V_{2,2}$ are recalculated using equation (4.3) to determine the values of $V_{1,1}$ and $V_{2,1}$ as follows:

$$V_{1,2} = (V_{1,3} - M_{2,2}) + M_{1,3} = (3-8)+5 = 0$$

$$V_{1,1} = (V_{1,2} - M_{2,1}) + M_{1,2} = (0-15)+20 = 5$$

$$V_{2,2} = (V_{2,3} - M_{3,2}) + M_{2,3} = (7-7)+12 = 12$$

$$V_{2,1} = (V_{2,2} - M_{3,1}) + M_{2,2} = (12-20)+8 = 0$$

Using the values of the variables and the required elements from the batch process recipes matrix, the makespan for the specified production sequence is calculated using equation (4.4) as follows:

$$Makespan = \sum_{j=1}^{m} M_{1,j} + \sum_{i=2}^{n} M_{i,m} + \sum_{i=1}^{n-1} V_{i,m} = 66 \text{ hours}$$

The makespan calculated above is for the production sequence specified by the matrix where product A is produced first, followed by product B and C as shown by the Gantt chart in Figure 4.4.

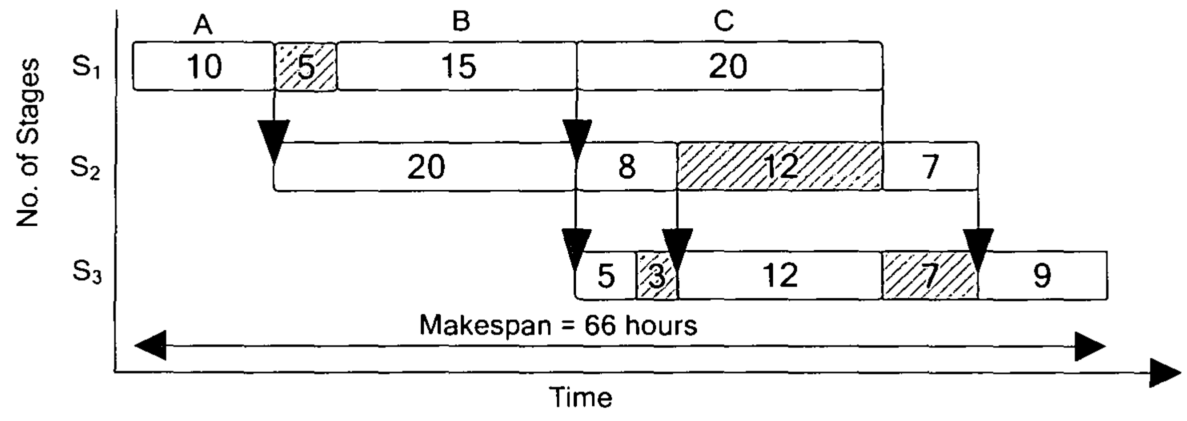


Figure 4.4: Gantt chart for production sequence ABC for example 4.3

4.2.1.2 NIS/UW transfer policy

The same batch scheduling problem used for the ZW which consists of three stages S_1, S_2 and S_3, with negligible transfer and setup time is used for the purpose of elaborating the makespan calculation for no intermediate storage policy. However, the batch process recipes have been changed to aid the illustrations better to reflect the NIS/UW.

Example 4.4

In this example, the makespan calculation is performed for three products i.e., A, B and C. The respective processing times for the batch process recipes producing three products in the sequence of product A, followed by B and lastly C, are shown in Table 4.5. It appears from Figure 4.5 that the possible number of paths that could be used to calculate the makespan is now increased compared to Example 4.4. The makespan calculated is 27 hours as shown in Figure 4.5 below.

Table 4.5
Processing time for production sequence ABC for example 4.4

Products	Processing time (hour)		
	S_1	S_2	S_3
A	5	8	6
B	9	3	2
C	4	5	3

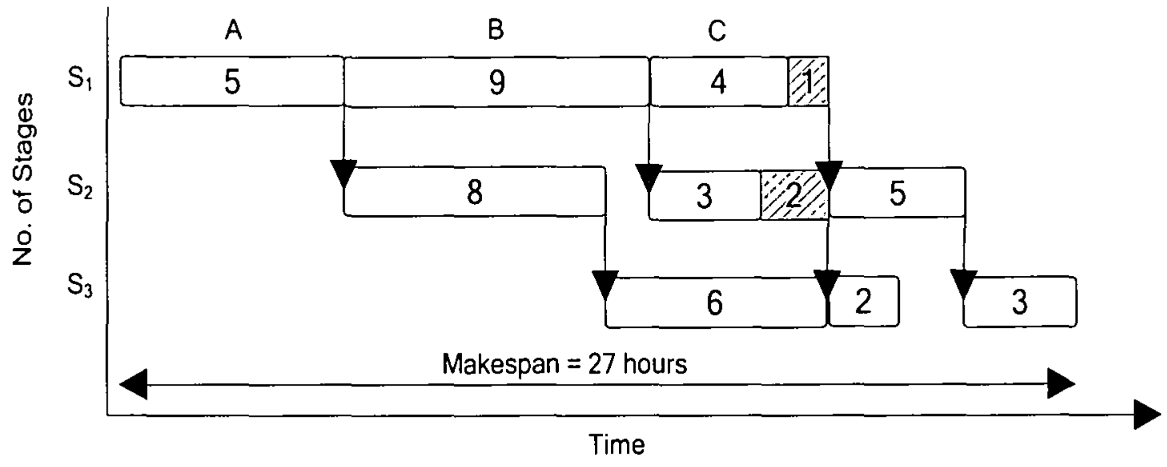


Figure 4.5: Gantt chart for production sequence ABC for example 4.4

Example 4.5

In this example, the makespan calculation is performed for a batch process producing four products according to the sequence of product A, followed by B, then C and finally D using three processing stages. The batch process recipes for the production of the four products are shown in Table 4.6. The observations made on the Gantt chart in Figure 4.6 show that the number of possible paths to calculate the makespan increases further with the increase in number of products. The makespan calculated is 31 hours as shown in Figure 4.6 below.

Table 4.6
Processing time for production sequence ABCD for example 4.5

Products	Processing time (hour)		
	S_1	S_2	S_3
A	5	8	6
B	9	3	2
C	4	5	3
D	4	5	2

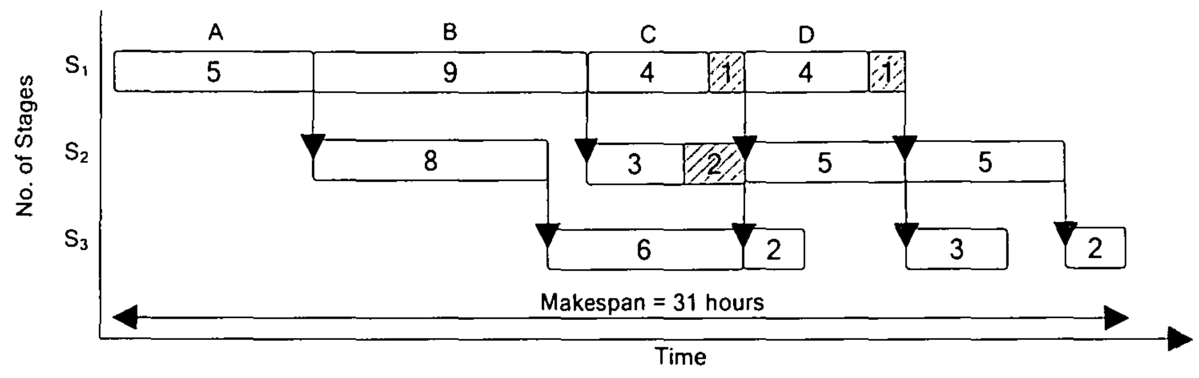


Figure 4.6: Gantt chart for production sequence ABCD for example 4.5

From the above observations, a conclusion similar to ZW transfer policy can be drawn. The number of paths used for calculating the makespan increases with the increased number of products but the calculated makespan remains the same regardless of the path selected. This is because the beginning and the end points for all the paths are located at the same spot. Therefore, a common path could again be selected for calculating the makespan for any batch process recipe similar to the case of ZW transfer policy.

Matrix Representation

The determination of makespan using matrix representation can be done by firstly representing the batch process recipe in the form of a matrix and to also include the variables as shown below. In this respect the scheduling would be based on a sequence where product A is produced first, followed by B, then C and lastly D.

$$\begin{array}{cccc}
 & 1 & 2 & 3 \\
1 & AS_1 & AS_2 & AS_3 \\
 & V_{1,1} & V_{1,2} & V_{1,3} \\
2 & BS_1 & BS_2 & BS_3 \\
 & V_{2,1} & V_{2,2} & V_{2,3} \\
3 & CS_1 & CS_2 & CS_3 \\
 & V_{3,1} & V_{3,2} & V_{3,3} \\
4 & DS_1 & DS_2 & DS_3
\end{array}$$

From the observations on the Gantt chart, it also appears that another variable is required in addition to the variable, V, to represent the holding time as indicated by the shaded area in the Gantt chart shown in Figure 4.5 and 4.6. This is due to the transfer policy adopted which allows product intermediate to remain within the same stage until the availability of the next one. For calculation purpose, this variable is represented by the letter 'I'.

Also, from the Gantt charts, it could easily be realized that the variables 'V' in between the first stage for all products i.e., $V_{1,1}$, $V_{2,1}$ and $V_{3,1}$ would always take a zero value. This is because, in the NIS/UW, the product intermediates are allowed to stay inside the same stage when the next stage is not yet available and therefore, the idle time is zero in between the first stages of all the products.

Further, it is also important to note that there would not be any holding time in all the process stages for the first product. This is because all the next stages would be available to take the product intermediate of the first product from the earlier stages. Similarly, there would not be any holding time in the last stages of all the products.

As such, the above matrix with the introduction of the variable 'I' could be rewritten as follows:

$$\begin{array}{cccc}
 & 1 & 2 & 3 \\
1 & AS_1 & AS_2 & AS_3 \\
 & V_{1,1} & V_{1,2} & V_{1,3} \\
2 & BS_1 \; I_{1,1} & BS_2 \; I_{1,2} & BS_3 \\
 & V_{2,1} & V_{2,2} & V_{2,3} \\
3 & CS_1 \; I_{2,1} & CS_2 \; I_{2,2} & CS_3 \\
 & V_{3,1} & V_{3,2} & V_{3,3} \\
4 & DS_1 \; I_{3,1} & DS_2 \; I_{3,2} & DS_3
\end{array}$$

For ease of calculation, the subscripts used for variable 'I' to represent the respective holding time will follow the similar arrangement made to represent the inter-stage idle times using variable 'V' in rows and columns of the above matrix. It is worth noting that the variable $I_{1,1}$ represents the holding time within the process stage BS_1, the variable $I_{1,2}$ represents the holding time within the stage BS_2 and so on.

The calculation of the variables 'V' can be done in a similar manner as in the case of ZW transfer policy with exception that there is no need to perform the reverse calculation. This is because, in NIS/UW, the product intermediate is allowed to remain within the same stage whenever the next stage is unavailable. Also it does not depend on the availability of succeeding stages as observed in the case of ZW transfer policy. For example, in Figure 4.6, the stage CS_1 holds the product intermediate for one hour until the stage CS_2 is made available but it does not depend on the availability of stage CS_3.

The determination of the variables, V, in the above matrix starts from the second stage between two consecutive products. This is because the variables 'V' located between the first stages of all the products are zero as explained earlier.

Firstly, the variable $V_{1,2}$ is calculated. This is followed by the calculation of variable $V_{1,3}$ which is based on the value of variable $V_{1,2}$ as shown below.

$V_{1,2} = BS_1 - AS_2$

$V_{1,3} = (V_{1,2} + BS_2) - AS_3$

The same procedure is repeated for calculating the other variables 'V' in the above matrix. However, compared to ZW transfer policy, the formula to calculate the succeeding variables 'V' in NIS/UW must also incorporate the additional variable 'I' which is used to represent the holding time inside the same stage, if required, as shown by the equations below.

$V_{2,2} = CS_1 - (BS_2 + I_{1,2})$

$V_{2,3} = (V_{2,2} + CS_2) - BS_3$

$V_{3,2} = DS_1 - (CS_2 + I_{2,2})$

$V_{3,3} = (V_{3,2} + DS_2) - CS_3$

It must be noted that if calculated value of the variable 'V' is negative, it would be assigned a zero value instead. This is to show that there is no idle time between process stages. Nevertheless, the calculated value represents the time duration for which the product intermediate must be held inside the earlier process stage i.e., the holding time, I. For example, in the Figure 4.6, the calculation of variable $V_{1,3}$ between AS_3 and BS_3

gives -2 hours. This shows that the idle time is zero but the product intermediate must be held for 2 hours in stage BS_2 before it is transferred to stage BS_3.

The makespan for the multi product batch process is calculated using the same path as adopted earlier in ZW transfer policy as follows.

$$Makespan = AS_1+AS_2+AS_3+V_{1,3}+BS_3+V_{2,3}+CS_3+V_{3,3}+DS_3$$

The generalized mathematical expression for the matrix representation

Similar to the ZW transfer policy, to apply the above procedure for more than four products and three stages, following equations are developed. The calculation procedure begins with the first two rows of the matrix and then carried forward to the succeeding rows using equations 4.5 and 4.6 respectively.

$$V_{i,1} = 0,\ I_{i,m} = 0 \qquad i = 1.......n\text{-}1$$

$$V_{1,j+1} = (V_{1,j} + M_{2,j}) - M_{1,j+1} \qquad j = 1........m-1 \tag{4.5}$$

$$V_{i,j+1} = (V_{i,j} + M_{i+1,j}) - (M_{i,j+1} + I_{i-1,j+1}) \qquad j = 1........m-1,\ i = 2.......n\text{-}1 \tag{4.6}$$

As explained above, if the calculated value of $V_{i,j+1}$ is negative, a zero value is assigned instead. Nevertheless, the calculated value sign will be changed to positive and assigned to $I_{i,j}$ i.e., holding time in the earlier stage. Otherwise, the value of $I_{i,j}$ will be zero.

The developed matrix formulation above is validated against an example problem taken from Ku and Karimi (1992) for which, they have developed an MILP formulation to solve for the optimum production sequence offering minimum makespan. Using the proposed method, the makespan is calculated.

Example 4.6

	1	2	3
1	3.5	4.3	8.7
	$V_{1,1}$	$V_{1,2}$	$V_{1,3}$
2	4.0 $I_{1,1}$	5.5 $I_{1,2}$	3.5
	$V_{2,1}$	$V_{2,2}$	$V_{2,3}$
3	3.5 $I_{2,1}$	7.5 $I_{2,2}$	6.0
	$V_{3,1}$	$V_{3,2}$	$V_{3,3}$
4	12.0 $I_{3,1}$	3.5 $I_{3,2}$	8.0

The variables 'V' and 'I' is determined using equations (4.5) and (4.6) as demonstrated below:

$V_{1,2} = M_{2,1} - M_{1,2} = -0.3 = 0 \quad , \quad I_{1,1} = 0.3$

$V_{1,3} = (V_{1,2} + M_{2,2}) - M_{1,3} = -3.2 = 0 \quad , \quad I_{1,2} = 3.2$

$V_{2,2} = M_{3,1} - (M_{2,2} + I_{1,2}) = -5.2 = 0 \quad , \quad I_{2,1} = 5.2$

$V_{2,3} = (V_{2,2} + M_{3,2}) - M_{2,3} = 4.0 \quad , \quad I_{2,2} = 0$

$V_{3,2} = M_{4,1} - (M_{3,2} + I_{2,2}) = 4.5 \quad , \quad I_{3,1} = 0$

$V_{3,3} = (V_{3,2} + M_{4,2}) - M_{3,3} = 2.0 \quad , \quad I_{3,2} = 0$

Subsequently, the makespan for the batch production sequence is calculated using the equation (4.4).

$$Makespan = \sum_{j=1}^{m} M_{1,j} + \sum_{i=2}^{n} M_{i,m} + \sum_{i=1}^{n-1} V_{i,m} = 40 \text{ hours}$$

Again it should be noted that the makespan calculated is based on the production sequence in the order of product A, followed by B, then C and lastly D as shown by the Gantt chart in Figure 4.7.

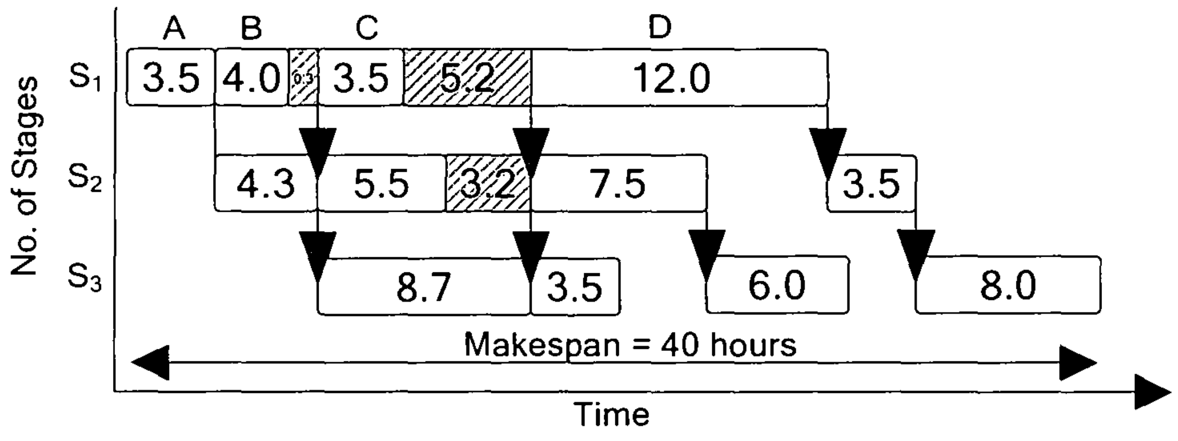


Figure 4.7: Gantt chart for production sequence ABCD for example 4.6

4.2.1.3 UIS/UW transfer policy

Often in a batch process, intermediate storage is used for temporary storage of product intermediate in between the process stages. This could be due to some process limitations where product intermediate is not allowed to stay within the same process stage after its processing and therefore, must be transferred to the temporary storage until the next stage is available. The number and location of the temporary storage required have a significant impact on the makespan during the batch process scheduling.

The assumptions made earlier for developing the makespan algorithm in the case of ZW and NIS/UW, are also applied for UIS/UW. However, additional assumptions are required specifically for this transfer policy and are listed below:

- In the event when the succeeding process stage is not yet available, the product intermediate will be transferred to the temporary storage immediately after the process is completed. Nevertheless, if the next process stage is available, the product intermediate is sent directly to the next stage.
- Storage time inside the temporary storage depends on the earliest availability of the next stage.
- The number and location of temporary storages are treated separately for each stage

Example 4.7

In this example, the makespan calculation is performed for three products i.e., A, B and C using the same method as demonstrated earlier. The batch process recipes for the three products based on three processing stages are shown in Table 4.7. As expected, the number of possible paths identified to calculate the makespan increases with increase in the number of products. The makespan calculated is 26 hours as illustrated in Figure 4.8 below.

Table 4.7

Processing time for production sequence ABC for example 4.7

Products	Processing time (hour)		
	S_1	S_2	S_3
A	5	8	6
B	6	5	2
C	3	5	3

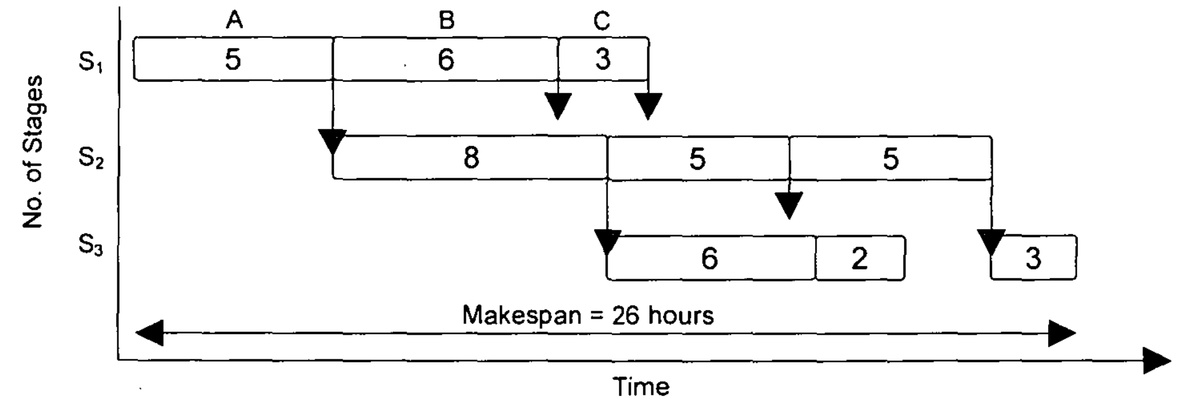


Figure 4.8: Gantt chart for production sequence ABC for example 4.7

It is also observed from Figure 4.8 that with increasing number of products and possible paths, the number of temporary storages also increases. In the present case, there are three locations identified where temporary storages are required.

The first location is after the processing of product B in stage S_1. This is due to the unavailability of stage S_2 where the processing of product A intermediate still takes place.

The second location is after the processing of product C in stage S_1. This is due to the unavailability of stage S_2 where the processing of product B intermediate still takes place.

The last location is after the processing of product B intermediate in stage S_2. This is due to the unavailability of stage S_3 where the processing of product A intermediate still takes place.

Finally, the makespan is determined using the path similar to the one used in example 4.8 i.e., by taking the sum of processing times of stages AS_1, AS_2, AS_3, BS_3 and CS_3. However, in this example, the makespan calculation must take into consideration the idle time that exists between stage BS_3 and CS_3 which is 2 hours i.e., $(BS_2+CS_2) - (AS_3+BS_3)$.

Example 4.8

In this example, the makespan calculation is performed for four products A, B, C and D. The batch process recipes for the four products based on three processing stages are shown in Table 4.8. It can be observed from Figure 4.9 that the number of possible paths identified to calculate the makespan has now increased further compared to example 4.7. The makespan calculated is 29 hours.

Table 4.8

Processing time for production sequence ABCD for example 4.8

Products	Processing time (hour)		
	S_1	S_2	S_3
A	5	8	6
B	6	5	2
C	3	5	3
D	3	4	2

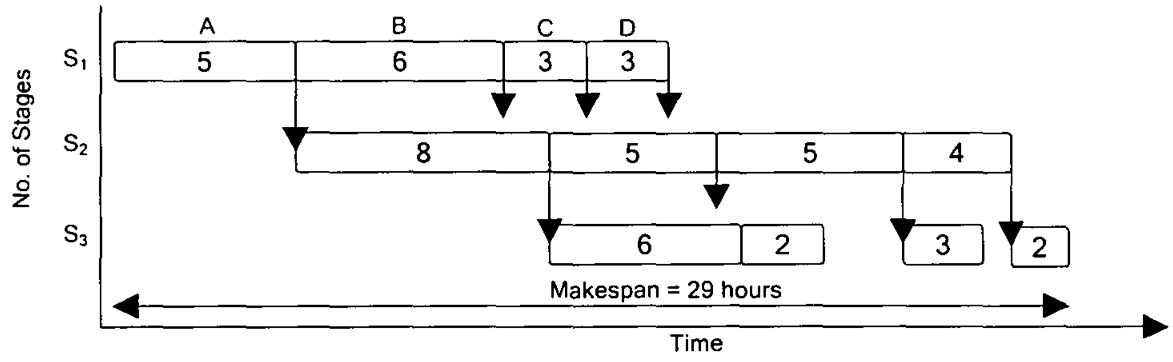


Figure 4.9: Gantt chart for production sequence ABCD for example 4.8

It is also observed from Figure 4.9 that the number of locations for temporary storage has also increased. There are four locations where temporary storages are needed.

The first location is after the processing of product B in stage S_1. This is due to the unavailability of stage S_2 which is still processing the product intermediate of A.

The second location is after processing of product C in stage S_1. This is due to the unavailability of stage S_2 which is still processing the product intermediate of B. The third location is after processing of product D in stage S_1. This is due to the unavailability of stage S_2 which is still processing the product intermediate of B followed by the processing of product intermediate of C.

The last location is after processing the product B intermediate in stage S_2. This is due to the unavailability of stage S_3 which is still processing product intermediate of A.

Finally, the makespan is determined using the same path as selected in the previous examples i.e., by taking the sum of AS_1, AS_2, AS_3, BS_3, CS_3 and DS_3. However, similar to example 4.7, the idle times that exist between stages BS_3 and CS_3, and, CS_3 and DS_3, have to be included also in the makespan calculation. In the present case, the idle time between stage BS_3 and CS_3 is 2 hours i.e., $(BS_2+CS_2) - (AS_3+BS_3)$ and between CS_3 and DS_3, is 1 hours i.e., (DS_2-CS_3).

Matrix Representation

The following steps describe in detail the development of the proposed method using matrix representation for the batch process recipe with unlimited intermediate storage transfer policy. The production sequence describing the four products with respective variables are arranged in the matrix as shown below. The production follows the sequence in which product A is produced first, followed consecutively by product B, then C and lastly D.

$$\begin{array}{cccc}
 & 1 & 2 & 3 \\
1 & AS_1 & AS_2 & AS_3 \\
 & V_{1,1} & V_{1,2} & V_{1,3} \\
2 & BS_1 & BS_2 & BS_3 \\
 & V_{2,1} & V_{2,2} & V_{2,3} \\
3 & CS_1 & CS_2 & CS_3 \\
 & V_{3,1} & V_{3,2} & V_{3,3} \\
4 & DS_1 & DS_2 & DS_3
\end{array}$$

From the observation on the Gantt chart, it is recognized that a new variable is required in addition to the variable 'V'. This new variable represents the waiting time of product intermediate in the temporary storage indicated by the hanging arrows in the Gantt charts shown in Figures 4.8 and 4.9. This is because the product intermediate could be stored temporarily in a temporary storage until the availability of the next stage for this case. For calculation purpose, this new variable is represented using the letter W.

Also from the Gantt chart, it appears that the variables 'V' located in between the first stage for all the products i.e., $V_{1,1}$, $V_{2,1}$ and $V_{3,1}$ would have a value of zero. This is because the product intermediates could be transferred immediately to the temporary storage in case of the unavailability of the next stage. Therefore, there will not be any idle time for the first stage for all the products as shown in Figure 4.9.

Also there is no requirement for temporary storage after the processing of the first product on all the process stages. This is due to the fact that all the process stages will always be available for processing the first product. Similarly, there will not be any temporary storage required after the last stage for all the products.

As such, the above matrix with the introduction of the variable ‘W’ could be rewritten as follows:

$$\begin{array}{cccc}
 & 1 & 2 & 3 \\
1 & AS_1 & AS_2 & AS_3 \\
 & V_{1,1} & V_{1,2} & V_{1,3} \\
2 & BS_1\ W_{1,1} & BS_2\ W_{1,2} & BS_3 \\
 & V_{2,1} & V_{2,2} & V_{2,3} \\
3 & CS_1\ W_{2,1} & CS_2\ W_{2,2} & CS_3 \\
 & V_{3,1} & V_{3,2} & V_{3,3} \\
4 & DS_1\ W_{3,1} & DS_2\ W_{3,2} & DS_3
\end{array}$$

For ease of calculation, the subscripts used for the variables ‘W’ to represent the waiting time, will follow the similar arrangement used to represent inter-stage idle times using variables ‘V’ in the respective rows and columns of the above matrix. However, it must be noted that the variable $W_{1,1}$ represents the waiting time after stage BS_1, the variable $W_{1,2}$ represents the waiting time after stage BS_2 and so on.

The calculation of variables can be done in a similar manner as in the case of NIS transfer policy. However, the only difference in the calculation procedure is that in the case of unavailability of the next stage, a temporary storage is always available in UIS/UW whereas for the NIS/UW, the product intermediate is held within the same stage.

Firstly, the variable $V_{1,2}$ is calculated. This is followed by the calculation of variable $V_{1,3}$ which is based on the value of variable $V_{1,2}$ as shown below.

$$V_{1,2} = BS_1 - AS_2$$

$$V_{1,3} = (V_{1,2} + BS_2) - AS_3$$

The same procedure is repeated for the calculation of all other variables 'V' in the above matrix. Compared to the procedure adopted for calculating the variable for ZW and NIS transfer policies, the formula for determining each of the succeeding variables 'V' in UIS/UW is different. In this case, the formula must also incorporate the processing times of all the previous products on respective stages in addition to all the previous variables 'V'. For example, in Figure 4.9, the product C is sent to the temporary storage after stage S_1 due to the unavailability of stage S_2 as it is still processing the product B intermediate. The storage time for product C intermediate in the temporary storage depends on the processing time of product B intermediate in stage S_2. The same applies to the case with product D intermediate. Product D is sent to the temporary storage after stage S_1 due to the unavailability of stage S_2 as it is still processing the product B intermediate followed by the processing of product C intermediate. The storage time for product D intermediate in the temporary storage now depends on the processing time of product B intermediate as well as the processing time of product C intermediate in stage S_2. In view of these observations, the formulae developed to calculate all the succeeding variables 'V' are as shown below.

$$V_{2,2} = (BS_1+CS_1) - (AS_2+V_{1,2}+BS_2)$$

$$V_{2,3} = (V_{1,2}+BS_2+V_{2,2}+ CS_2) - (AS_3+V_{1,3}+BS_3)$$

$$V_{3,2} = (BS_1+CS_1+DS_1) - (AS_2+V_{1,2}+BS_2+V_{2,2}+CS_2)$$

$$V_{3,3} = (V_{1,2}+BS_2+V_{2,2}+CS_2+V_{3,2} + DS_2) - (AS_3+V_{1,3}+BS_3+V_{2,3}+CS_3)$$

It must be noted that if the calculated values of the variable 'V' is negative, a zero value will be assigned instead. The negative value indicates that the idle time is zero between the respective process stages. However, the calculated value represents the time duration for which the product intermediate must be stored in a temporary storage until the availability of the next process stage. For example, in Figure 4.9, the calculation of variable $V_{1,2}$ between AS_2 and BS_2 gives -2 hours. This indicates that there is no idle time between the two process stages. However, the product intermediate has to be stored for 2 hours in a temporary storage made available after stage BS_1 as indicated by the hanging arrow in Figure 4.9.

The makespan for the multi product batch process is calculated using the same path as adopted earlier in ZW and NIS transfer policies as follows:

$$Makespan = AS_1+AS_2+AS_3+V_{1,3}+ BS_3+ V_{2,3}+CS_3+V_{3,3}+DS_3$$

The generalized mathematical expression for the matrix representation

Likewise ZW and NIS transfer policies, the following equations have been developed for "n" number of products to be processed in "m" number of stages. The calculation procedure begins between the first two rows of the matrix and then carried forward to the succeeding rows using equations 4.7 and 4.8 respectively.

$$V_{i,1} = 0 \qquad i = 1......n-1$$

$$V_{1,j} = (M_{2,j-1} + V_{1,j-1}) - M_{1,j} \qquad j = 2......m \qquad (4.7)$$

$$V_{i,j} = (\sum_{k=2}^{i+1} M_{k,j-1} + \sum_{k=1}^{i} V_{k,j-1}) - (\sum_{k=1}^{i} M_{k,j} + \sum_{k=1}^{i-1} V_{k,j}) \quad j = 2.......m, \qquad i = 2........n-1 \qquad (4.8)$$

If the calculated values of $V_{1,j}$ and $V_{i,j}$ are negative, a zero value is assigned instead. However, the calculated values will represent the time needed to store the product intermediate in a temporary storage.Therefore, the calculated value signs will be changed to positive and assigned to $W_{1,j-1}$ and $W_{i,j-1}$ respectively i.e., waiting time inside the temporary storage.Otherwise, the values of $W_{1,j-1}$ and $W_{i,j-1}$ will be zero.

The method is verified by applying it on the same process recipe from Table 4.8 as shown below:

$$\begin{array}{cccc}
 & 1 & 2 & 3 \\
1 & 5 & 8 & 6 \\
 & V_{1,1} & V_{1,2} & V_{1,3} \\
2 & 6\ \ W_{1,1} & 5\ \ W_{1,2} & 2 \\
 & V_{2,1} & V_{2,2} & V_{2,3} \\
3 & 3\ \ W_{2,1} & 5\ \ W_{2,2} & 3 \\
 & V_{3,1} & V_{3,2} & V_{3,3} \\
4 & 3\ \ W_{3,1} & 4\ \ W_{3,2} & 2
\end{array}$$

The matrix above shows the arrangement of the batch process recipes according to the selected production sequence of A, B, C and D. The calculations for the variables using equation (4.7) and (4.8) are shown below.

$$V_{1,2} = (M_{2,1} - M_{1,2}) = -2 = 0 \qquad , \quad W_{1,1} = 2$$

$$V_{1,3} = (M_{2,2}+V_{1,2}) - M_{1,3} = -1 = 0 \qquad , \quad W_{1,2} = 1$$

$$V_{2,2} = (M_{2,1}+M_{3,1}) - (M_{1,2}+M_{2,2}+V_{1,2}) = -4 = 0 \qquad , \quad W_{2,1} = 4$$

$$V_{2,3} = (M_{2,2}+M_{3,2}+V_{1,2}+V_{2,2}) - (M_{1,3}+M_{2,3}+V_{1,3}) = 2 \qquad , \quad W_{2,2} = 0$$

$$V_{3,2} = (M_{2,1}+M_{3,1}+M_{4,1}) - (M_{1,2}+M_{2,2}+M_{3,2}+V_{1,2}+V_{2,2}) = -6 = 0 \qquad , \quad W_{3,1} = 6$$

$V_{3,3} = (M_{2,2}+M_{3,2}+M_{4,2}+V_{1,2}+V_{2,2}+V_{3,2}) - (M_{1,3}+M_{2,3}+M_{3,3}+V_{1,3}+V_{2,3}) = 1$, $W_{3,2} = 0$

Using the results obtained above, the makespan is calculated using equation (4.4).

$$Makespan = \sum_{j=1}^{m} M_{1,j} + \sum_{i=2}^{n} M_{i,m} + \sum_{i=1}^{n-1} V_{i,m} = 29 \text{ hours}$$

Note that the result obtained is equal to the result obtained from the Gantt chart method as demonstrated in Example 4.8.

Number of Temporary Storages

Next, the number of temporary storages required is determined. From the calculation of variables above, it is recognized that the calculated values of variables $W_{1,1}$, $W_{2,1}$, $W_{3,1}$ and $W_{1,2}$ are greater than 0. This indicates that there is requirement of four temporary storages in total i.e., three after first stage and one after second stage as shown in Figure 4.10. The calculated values of these variables represent the duration of product intermediate inside these temporary storages. This is also in accordance with the Gantt chart shown in Figure 4.9 for the same batch process recipe.

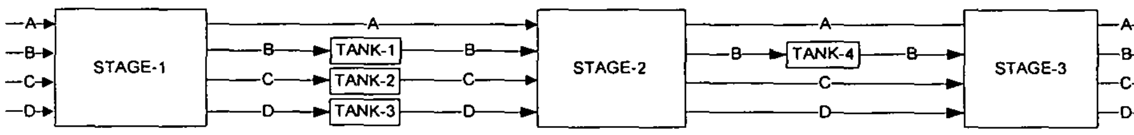


Figure 4.10: Temporary storage arrangement for production sequence ABCD for example 4.8

4.2.1.4 FIS/UW transfer policy

From the UIS/UW, it can be deduced that the temporary storages are used as transfer media between process stages to reduce the idle time and finally the process makespan. However, the provision of unlimited temporary storages in UIS/UW would increase the overall production cost. For economic production requirement, the concept of limited temporary storages in a batch process is introduced and is known as finite intermediate storage (FIS) or limited intermediate storage (LIS) (Pitty and Karimi, 2008). The FIS/UW is usually applied with NIS/UW (Kim et al. 1996).

In FIS/UW, the number of temporary storages is limited. In the case of unavailability of the next process stage or temporary storage for product intermediate of succeeding product, NIS/UW allows the product to remain in the same stage.

The FIS/UW offers the flexibility of using the limited storage for all products in a batch process. The number of storages is limited and predefined for any product recipe in a batch process. For demonstration purpose, the following assumptions for FIS/UW are made with few exceptions to the assumptions made earlier in the cases of ZW, NIS/UW and UIS/UW transfer policies.

1. If the downstream processing stage or temporary storage is not available, the product intermediate of the succeeding product has to remain in the same processing stage. In this context the NIS/UW is adopted.

2. It is assumed that there is a dedicated temporary storage available between every two stages i.e. it would not be shared among process stages.

Example 4.9

This example performs the makespan calculation for three products i.e., A, B and C using the same method as discussed earlier. The processing times for the three products based on three processing stages are shown in Table 4.9. The makespan calculated is 24 hours as illustrated in Figure 4.11.

Table 4.9
Processing time for production sequence ABC for example 4.9

Products	Processing time (hour)		
	S_1	S_2	S_3
A	4	10	5
B	3	2	3
C	5	2	2

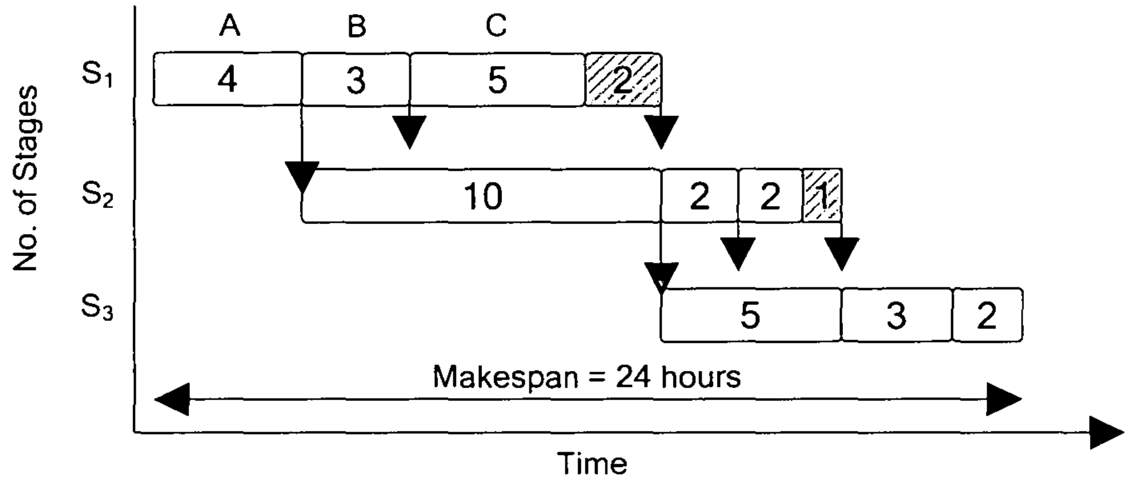


Figure 4.11: Gantt chart for production sequence ABC for example 4.9

It is also observed from Figure 4.11 that the number of locations, where temporary storage is required, increases with increasing number of products. As such, in Figure 4.11, there are four locations where temporary storages are required.

The first location is after the processing of product B in stage S_1. This is due to the unavailability of stage S_2 as it is still processing the product A intermediate. The time for storage of product B intermediate after stage S_1 is 7 hours i.e., AS_2-BS_1.

The second location is after the processing of product C in stage S_1. This is due to the unavailability of stage S_2 as it is still processing the product A intermediate which is followed by the processing of product B intermediate. Also, at this point of time, the temporary storage available after stage S_1 is storing the product B intermediate. Therefore, the product C intermediate must remain in the stage S_1 until the temporary storage is available i.e., NIS/UW is adopted after the processing of product C in stage S_1 as shown by the shaded area in Figure 4.11. The holding time of product C intermediate in the stage S_1 is 2 hours and is calculated by subtracting the processing time of the product C intermediate in the first stage S_1 from the storage time of the product B intermediate in the temporary storage. From Figure 4.11, the time of storage of product C intermediate after stage S_1 in the temporary storage on its availability is calculated as 2 hours.

The third location is after the processing of product B intermediate in stage S_2. This is due to the unavailability of stage S_3 as it is still processing the product A intermediate. The storage time calculated for the temporary storage of product B intermediate from Figure 4.11 is 3 hours i.e., AS_3-BS_2.

The last location is after the processing of product C intermediate in stage S_2. This is due to the unavailability of stage S_3 as it is still processing the product A intermediate which is followed by the processing of product B intermediate. Similar to the case of availability of temporary storage after stage 1, at this point of time, the temporary storage available after stage S_2 is still storing the product B intermediate. Therefore, the product C intermediate must remain in the stage S_2 until the temporary storage is available i.e., NIS/UW is adopted after the processing of product C intermediate in stage S_2 as shown by the shaded area in Figure 4.11. The holding time of product C intermediate in stage S_2 is 1 hour and is calculated by subtracting the processing time of product C intermediate in stage S_1 from the storage time of product B intermediate in the temporary storage. From Figure 4.11, the time of storage of product C intermediate after stage S_2 in the temporary storage on its availability is calculated as 3 hours.

Finally, the makespan of the batch process is calculated using the same path selected earlier as common path from all the possible identified paths i.e., by taking the sum of AS_1, AS_2, AS_3, BS_3 and CS_3.

Example 4.10

This example performs the makespan calculation for four products A, B, C and D. The processing times for the four products based on three processing stages are shown in Table 4.10. The makespan calculated is 33 hours as illustrated in Figure 4.12.

Table 4.10
Processing time for production sequence ABCD for example 4.10

Products	Processing time (hour)		
	S_1	S_2	S_3
A	4	10	5
B	12	4	7
C	3	3	4
D	2	2	2

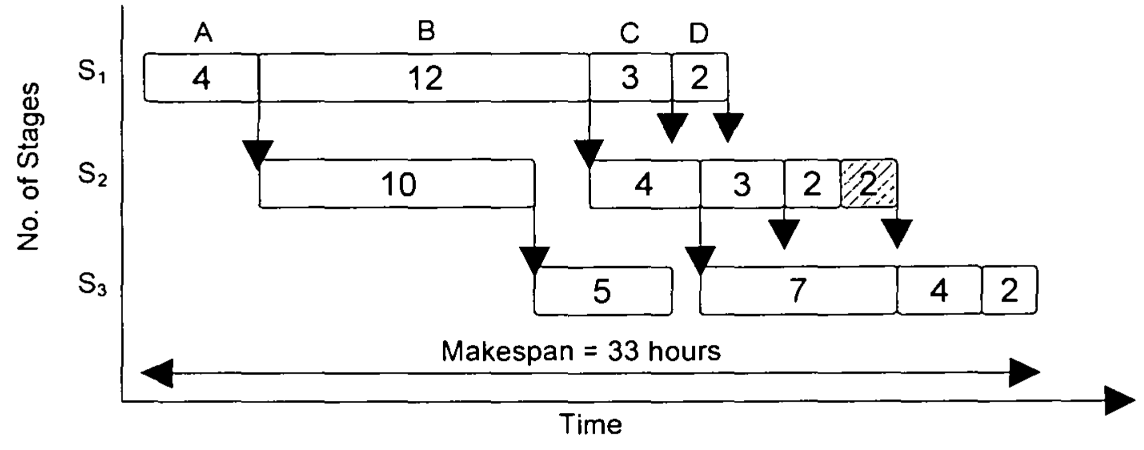


Figure 4.12: Gantt chart for production sequence ABCD for example 4.10

The comparison of Figure 4.11 and 4.12 shows the same number of locations where temporary storages are required i.e., four locations in both the cases. However, the number of times for which the NIS/UW is adopted in Figure 4.12 is less compared to the batch process shown in Figure 4.11 as shown by the shaded area in both the figures.

The first location is after the processing of product C in stage S_1. This is due to the unavailability of stage S_2 as it is still processing the product B intermediate. The time for storage of product C intermediate after stage S_1 from Figure 4.12 is 1 hours i.e., BS_2-CS_1.

The second location is after the processing of product D in stage S_1. This is due to the unavailability of stage S_2 as it is still processing the product C intermediate. From Figure 4.12, the time of storage of product D intermediate after stage S_1 in the temporary storage is calculated as 2 hours i.e., $(BS_2+CS_2) - (CS_1+DS_1)$.

The third location is after the processing of product C intermediate in stage S_2. This is due to the unavailability of stage S_3 as it is still processing the product B intermediate. The storage time calculated for the temporary storage of product C intermediate after stage S_2 from Figure 4.12 is 4 hours i.e., BS_3-CS_2.

The last location is after the processing of product D intermediate in stage S_2. This is due to the unavailability of stage S_3 as it is still processing the product B intermediate which is followed by the processing of product C intermediate. At this point of time, the temporary storage available after stage S_2 is still storing the product C intermediate. Therefore, the product D intermediate must remain in the stage S_2 till the temporary storage is available i.e., NIS/UW is adopted after the processing of product D intermediate in stage S_2 as shown by the shaded area in Figure 4.12. The holding time of product D intermediate in stage S_2 is 2 hour and is calculated by subtracting the processing time of product D intermediate in stage S_2 from the storage time of product C intermediate in the temporary storage. From Figure 4.12, the time of storage of product D intermediate after stage S_2 in the temporary storage on its availability is calculated as 4 hours.

Note that the makespan for the batch sequence can be calculated using the same path as adopted in the above examples i.e., by taking the sum of AS_1, AS_2, AS_3, BS_3, CS_3 and DS_3. However, there exists idle time between stages AS_3 and BS_3 that must also be included in makespan calculation. In this case the idle time between stages AS_3 and BS_3 is 1 hour i.e., $(BS_1+BS_2) - (AS_2+AS_3)$.

Matrix Representation

The developed matrix method for batch processes with finite intermediate storage is explained in the following steps. Note that A, B, C and D represent the products while S_1, S_2 and S_3 represent the stages with processing times according to the process sequence for producing each product. In this respect the scheduling will be based on a sequence where product A is produced first, followed by B, then C and lastly D.

$$\begin{array}{cccc} & 1 & 2 & 3 \\ 1 & AS_1 & AS_2 & AS_3 \\ & V_{1,1} & V_{1,2} & V_{1,3} \\ 2 & BS_1 & BS_2 & BS_3 \\ & V_{2,1} & V_{2,2} & V_{2,3} \\ 3 & CS_1 & CS_2 & CS_3 \\ & V_{3,1} & V_{3,2} & V_{3,3} \\ 4 & DS_1 & DS_2 & DS_3 \end{array}$$

From the observation on the Gantt charts, it appears that two additional variables are required in addition to the variable 'V'. These additional variables are used to represent the waiting time in a temporary storage illustrated by hanging arrows and the holding time inside the same stage illustrated by shaded areas in the Gantt charts. For calculation purpose, the letters, W and I, are used to represent these variables for the waiting and the holding time respectively.

It is also observed from the Gantt charts that the variables 'V' for the first stage of all the products would have zero value because there would not be any idle time for the first stage of all the products. This is because the processing of the next product in the first stage could be started immediately after the processing of the earlier product. This is in accordance with the assumptions made earlier for FIS/UW i.e., the product intermediate could immediately be transferred to the temporary storage or allowed to remain inside the same stage in case of unavailability of the next stage.

Also, it is obvious from the Gantt charts that there would not be any requirement of temporary storage or holding the product intermediate inside the same stage after processing the product intermediate of first product in all stages. This is because all stages will always be available for processing the product intermediate of the first product. Similarly, there would not be any requirement of holding the product intermediate of second product inside the same stage after its processing in all the stages. This is because a temporary storage will always be available for the product intermediate of the second product in case it is required.

Further to note that there will not be any requirement of temporary storage and holding of product intermediate inside the same stage for all the products after their processing in the last stages.

As such, the above matrix with the introduction of new variables 'W' and 'I' could be rewritten as follows:

$$\begin{array}{cccc}
 & 1 & 2 & 3 \\
1 & AS_1 & AS_2 & AS_3 \\
 & V_{1,1} & V_{1,2} & V_{1,3} \\
2 & BS_1\ W_{1,1}, I_{1,1} & BS_2\ W_{1,2}, I_{1,2} & BS_3 \\
 & V_{2,1} & V_{2,2} & V_{2,3} \\
3 & CS_1\ W_{2,1}, I_{2,1} & CS_2\ W_{2,2}, I_{2,2} & CS_3 \\
 & V_{3,1} & V_{3,2} & V_{3,3} \\
4 & DS_1\ W_{3,1}, I_{3,1} & DS_2\ W_{3,2}, I_{3,2} & DS_3
\end{array}$$

For ease of calculation, the subscripts used for the variables 'W' and 'I' will follow similar arrangement to represent variable 'V' in rows and columns of the above matrix. However, as indicated in the above matrix, the variables $W_{1,1}$ and $I_{1,1}$, represent the waiting and holding time for product intermediate of second product respectively after stage BS_1. Likewise, the variables, $W_{1,2}$ and $I_{1,2}$, represent the waiting and holding time for product intermediate of second product respectively after stage BS_2. For succeeding products, the arrangement of variables, W and I, is also shown in the above matrix.

The calculation of variables can be done in a similar manner to the UIS/UW case. However, in FIS/UW, the formulae for calculating the variable 'V' must also incorporate the variable 'I' to represent the holding time of product intermediates inside the same stage. This is to show that NIS/UW will be adopted in the case where the next stage and temporary storage are not available.

Firstly, the variable $V_{1,2}$ is calculated. This is followed by the calculation of $V_{1,3}$ which is based on the value of $V_{1,2}$ as shown below.

$$V_{1,2} = BS_1 - AS_2$$

$$V_{1,3} = (V_{1,2} + BS_2) - AS_3$$

The same procedure is repeated for the calculation of all other variables 'V' mentioned in the above matrix with the addition of variables 'I' to represent the holding time in the same stage as shown below.

$$V_{2,2} = (BS_1+I_{1,1}+CS_1) - (AS_2+V_{1,2}+BS_2+I_{1,2})$$

$$V_{2,3} = (V_{1,2}+BS_2+I_{1,2}+V_{2,2} + CS_2) - (AS_3+V_{1,3}+BS_3)$$

$$V_{3,2} = (BS_1+I_{1,1}+CS_1+I_{2,1}+DS_1) - (AS_2+V_{1,2}+BS_2+I_{1,2}+V_{2,2}+CS_2+I_{2,2})$$

$$V_{3,3} = (V_{1,2}+BS_2+I_{1,2}+V_{2,2}+CS_2+I_{2,2}+V_{3,2}+ DS_2) - (AS_3+V_{1,3}+BS_3+V_{2,3}+CS_3)$$

It must be noted that in case of any negative value obtained for variable 'V', a zero value is assigned instead. The negative value shows that the idle time is zero between process stages. However, the calculated value indicates the time duration for which the product intermediate must be stored in a temporary storage until availability of the next stage. Therefore, the calculated value will be assigned to the variable 'W'. For example, in the Figure 4.12, the calculation of variable $V_{2,3}$ between BS_3 and CS_3 gives -4 hours. This shows that the idle time is zero and the product intermediate has to be stored in a temporary storage for 4 hours after stage CS_2.

In case, the temporary storage as well as next stage is not available, the product intermediate has to remain inside the same stage and that is shown by introducing the variable 'I' in the above formulations. For example, in the Figure 4.12, the product D intermediate after stage S_2 remains in the stage S_2 for 2 hours until the temporary storage is made available.

The makespan for the multi product batch process is calculated using the same path as earlier adopted in other transfer policies as follows:

$$Makespan = AS_1+AS_2+AS_3+V_{1,3}+BS_3+ V_{2,3}+ CS_3+V_{3,3}+DS_3$$

The generalized mathematical expression for the matrix representation

Similar to other transfer policies, to apply the above procedure for "n" number of products and "m" number of stages, following equations have been developed. The calculation procedure begins between the first two rows of the matrix and then carried forward to the succeeding rows using equations (4.9) and (4.10) respectively.

$$V_{i,1} = 0, \quad I_{1,j} = 0 \qquad\qquad i = 1.........n\text{-}1, \quad j = 1.........m$$

$$I_{i,m} = 0 \qquad\qquad i = 2.........n\text{-}1$$

$$V_{1,j} = (M_{2,j-1} + V_{1,j-1}) - M_{1,j} \qquad\qquad j = 2........m \tag{4.9}$$

$$\left[\begin{array}{l} if \;\; (M_{i+1,j-1} + V_{i,j-1}) \le W_{i-1,j-1},\; I_{i,j-1} = W_{i-1,j-1} - (M_{i+1,j-1} + V_{i,j-1}) \;\; else\; I_{i,j-1} = 0 \\ \\ V_{i,j} = (\sum_{k=2}^{i+1} M_{k,j-1} + \sum_{k=1}^{i} V_{k,j-1} + \sum_{k=1}^{i} I_{k,j-1}) - (\sum_{k=1}^{i} M_{k,j} + \sum_{k=1}^{i-1} V_{k,j} + \sum_{k=1}^{i-1} I_{k,j}) \\ \\ \qquad\qquad\qquad\qquad j = 2......m,\; i = 2......n\text{-}1 \end{array}\right] \tag{4.10}$$

If the calculated value of $V_{1,j}$ and $V_{i,j}$ are negative, a value of zero will be assigned instead. Nevertheless, the calculated value signs will be changed to positive and assigned to $W_{1,j-1}$ and $W_{i,j-1}$ respectively i.e., waiting time inside the temporary storage. Otherwise, $W_{1,j-1}$ and $W_{i,j-1}$ will be zero.

The validation of the method is done by calculating the makespan for three stage multi product batch process for the production sequence of A being the first, followed by B, then C and finally D. For this purpose, the batch process is taken from Table 4.10 and is shown below:

	1	2	3
1	4	10	5
	$V_{1,1}$	$V_{1,2}$	$V_{1,3}$
2	12 $W_{1,1}$, $I_{1,1}$	4 $W_{1,2}$, $I_{1,2}$	7
	$V_{2,1}$	$V_{2,2}$	$V_{2,3}$
3	3 $W_{2,1}$, $I_{2,1}$	3 $W_{2,2}$, $I_{2,2}$	4
	$V_{3,1}$	$V_{3,2}$	$V_{3,3}$
4	2 $W_{3,1}$, $I_{3,1}$	2 $W_{3,2}$, $I_{3,2}$	2

The matrix above shows the arrangement of the batch process recipes according to the selected production sequence of A, B, C and D. The calculations for the variables using equation (4.9) and (4.10) are shown below.

$V_{1,2} = M_{2,1} - M_{1,2} = 2$, $W_{1,1}=0$,

$V_{1,3} = (M_{2,2}+V_{1,2}) - M_{1,3} = 1$, $W_{1,2}=0$,

$I_{2,1} = 0$

$V_{2,2} = (M_{2,1}+M_{3,1}+I_{1,1}+I_{2,1}) - (M_{1,2}+M_{2,2}+V_{1,2}+I_{1,2}) = -1 = 0$, $W_{2,1} = 1$

$I_{2,2} = 0$

$V_{2,3} = (M_{2,2}+M_{3,2}+V_{1,2}+V_{2,2}+I_{1,2}+I_{2,2}) - (M_{1,3}+M_{2,3}+V_{1,3}+I_{1,3}) = -4 = 0$, $W_{2,2} = 4$

$I_{3,1} = 0$

$V_{3,2} = (M_{2,1}+M_{3,1}+M_{4,1}+I_{1,1}+I_{2,1}+I_{3,1}) - (M_{1,2}+M_{2,2}+M_{3,2}+V_{1,2}+V_{2,2}+I_{1,2}+I_{2,2}) = -2 = 0$

, $W_{3,1}=2$

$I_{3,2} = 2$

$V_{3,3}=(M_{2,2}+M_{3,2}+M_{4,2}+V_{1,2}+V_{2,2}+V_{3,2}+I_{1,2}+I_{2,2}+I_{3,2})-(M_{1,3}+M_{2,3}+M_{3,3}+V_{1,3}+V_{2,3}+I_{1,3}+I_{2,3})$

$= -4 = 0$

, $W_{3,2}=4$

Using the results obtained above, the makespan is calculated using equation (4.4).

$$Makespan = \sum_{j=1}^{m} M_{1,j} + \sum_{i=2}^{n} M_{i,m} + \sum_{i=1}^{n-1} V_{i,m} = 33 \text{ hours}$$

Note that the result obtained is equal to the result obtained from the Gantt chart method as demonstrated in Example 4.10.

Figure 4.13 illustrates the configuration of the batch process with the production sequence ABCD. The small shaded box inside stage-2 illustrates the holding time of product D (i.e., $I_{3,2}$ = 2hr) after completion due to unavailability of temporary storage 'Tank-2' which at that point of time is storing product C intermediate.

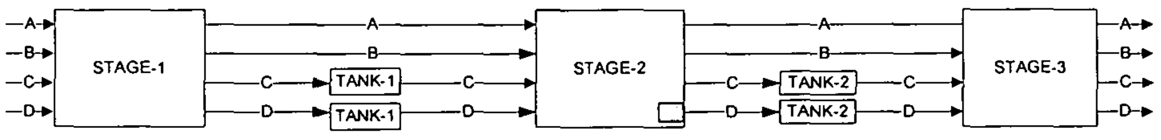


Figure 4.13: Temporary storage arrangement for production sequence ABCD for example 4.10

4.2.1.5 MIS/UW transfer policy

In MIS/UW, any of the transfer policies discussed earlier can be used in between the process stages. For instance, in a 4 stage batch process, if the process conditions require the product intermediate to wait in first stage until second stage becomes available and then wait in second stage until third stage becomes available then NIS/UW is followed. However, if the third stage requires the transfer of the product intermediate immediately to the fourth stage and fourth stage is not available then there is a need of temporary storage to keep the product intermediate till the fourth stage becomes available. In such cases, UIS/UW or FIS/UW could be adopted. Both of these transfer policies provide the temporary storages. However, UIS/UW is more flexible than FIS/UW. UIS/UW provides unlimited storage while FIS/UW provides limited storage. For demonstration purpose in this work, the application of MIS/UW has been discussed using NIS/UW and UIS/UW. For the purpose of understanding, the Gantt chart in the forthcoming example shows the way in which MIS/UW is operated.

Example 4.11

In this example, the makespan calculation is performed for three products i.e., A, B and C which are produced by following the sequence in which A is produced first, followed by B and lastly C, as shown in Table 4.11. The makespan calculated is 30 hours as shown in Figure 4.14 below.

Table 4.11
Processing time for production sequence ABC for example 4.11

Products	Processing time (hour)			
	S_1	S_2	S_3	S_4
A	5	8	6	4
B	6	4	2	2
C	6	5	3	3

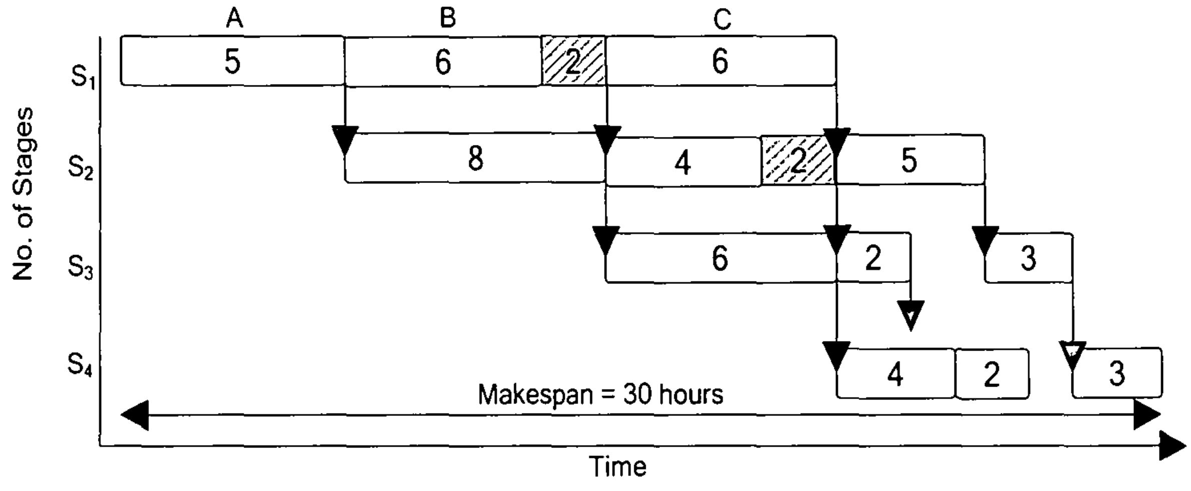


Figure 4.14: Gantt chart for production sequence ABC for example 4.11

It can be observed from Figure 4.14 that product B intermediate waits inside the stage 1 for 2 hours until the stage 2 becomes available i.e. NIS/UW is adopted. Same is the case when processing of product B intermediate completes in stage 2 i.e. it is again held for 2 hours until the stage 3 becomes available. However, same situation is not observed after processing of product B in stage 3. Here, it is again required to wait because stage 4 is not yet available but process conditions do not allow waiting inside the stage 3. Therefore, an alternate way is adopted in terms of providing a temporary storage to store the product B intermediate temporarily after stage 3 until the stage 4 becomes available. In other words, UIS/UW is adopted after stage 3.

Example 4.12

In this example, the makespan calculation is performed for a batch process producing four products according to the sequence of product A, followed by B, then C and finally D using four processing stages. The batch process recipes for the production of the four products are shown in Table 4.12. The makespan calculated is 33 hours as shown in Figure 4.15 below.

Table 4.12

Processing time for production sequence ABCD for example 4.12

Products	Processing time (hour)			
	S_1	S_2	S_3	S_4
A	5	8	6	4
B	9	3	2	2
C	4	5	3	4
D	4	4	2	2

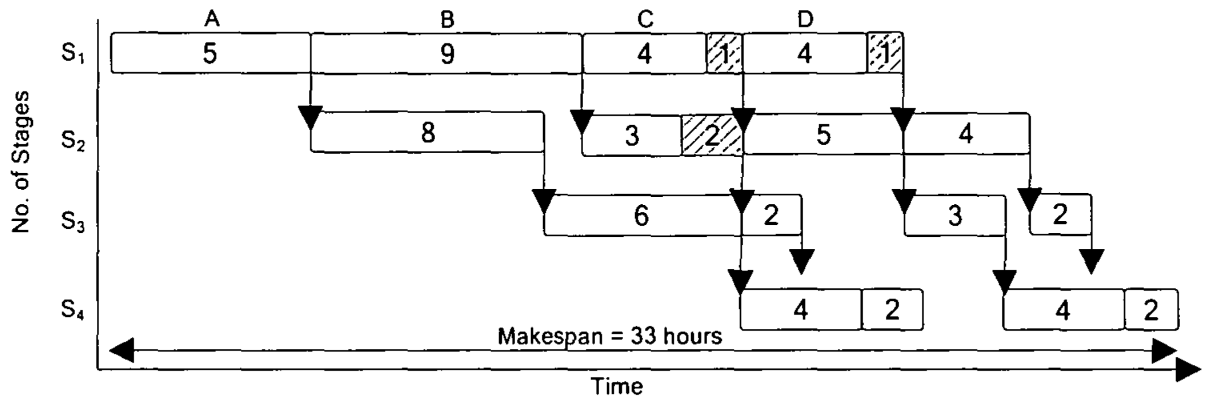


Figure 4.15: Gantt chart for production sequence ABCD for example 4.12

Figure 4.15 shows the observation similar to Figure 4.14 for the case of products C and D intermediates. In the present case, the product C intermediate is firstly held inside the stage 1 for 1 hour and then in stage 2 for 2 hours. However, as mentioned earlier, a temporary storage is provided after stage 3 to store the product C intermediate till the stage 4 becomes available. In case of product D, intermediate waits inside stage 1 for 1 hour and then the need of temporary storage arises after stage 3 to store the product D intermediate temporarily till the stage 4 becomes available.

The observations from above examples conclude that demand of storage depends upon the product recipe and based on that, the suitable transfer policy either NIS/UW or UIS/UW is adopted according to the predefined process conditions.

Makespan Determination

As stated above, for demonstration purpose, application of MIS/UW has been shown in line with NIS/UW and UIS/UW. Hence, the same matrix approach used earlier to develop mathematical formulations for determination of makespan in case of NIS/UW and UIS/UW could be employed for MIS/UW. As such the equations (4.5), (4.6), (4.7) and (4.8) have been re-written to determine the inter-stage idle times, holding time within same stage in case of NIS/UW and waiting time in temporary storages in case of UIS/UW. The equations could be modified with regard to their application between specific stages e.g. for example 4.12, equation 4.5 and 4.6 would be used for stages 1 and 2 while equation 4.7 and 4.8 for stage 3.

$$V_{i,1} = 0, \quad I_{i,m} = 0 \qquad i = 1 \ldots\ldots. n\text{-}1$$

$$V_{1,j+1} = (V_{1,j} + M_{2,j}) - M_{1,j+1} \qquad j = 1 \ldots\ldots.. m - 2 \qquad (4.5a)$$

$$V_{i,j+1} = (V_{i,j} + M_{i+1,j}) - (M_{i,j+1} + I_{i-1,j+1}) \qquad j = 1 \ldots\ldots.. m - 2, \quad i = 2 \ldots\ldots. n\text{-}1 \qquad (4.6a)$$

$$V_{1,j} = (M_{2,j-1} + V_{1,j-1}) - M_{1,j} \qquad j = m \qquad (4.7a)$$

$$V_{i,j} = (\sum_{k=2}^{i+1} M_{k,j-1} + \sum_{k=1}^{i} V_{k,j-1}) - (\sum_{k=1}^{i} M_{k,j} + \sum_{k=1}^{i-1} V_{k,j}) \quad j = m, \qquad i = 2........n-1 \qquad (4.8a)$$

The batch process recipe from example 4.12 would be used to demonstrate the application of above equations to determine makespan. The respective variables to represent the idle time 'V', holding time 'I' and waiting time 'W' are also shown in the below matrix.

$$\begin{array}{ccccc}
 & \multicolumn{3}{c}{\overbrace{\qquad\qquad\text{NIS/UW}\qquad\qquad}} & \\
 & & & \multicolumn{2}{c}{\overbrace{\qquad\text{UIS/UW}\qquad}} \\
 & 1 & 2 & 3 & 4 \\
1 & 5 & 8 & 6 & 4 \\
 & V_{1,1} & V_{1,2} & V_{1,3} & V_{1,4} \\
2 & 9\ I_{1,1} & 3\ I_{1,2} & 2\ W_{1,3} & 2 \\
 & V_{2,1} & V_{2,2} & V_{2,3} & V_{2,4} \\
3 & 4\ I_{2,1} & 5\ I_{2,2} & 3\ W_{2,3} & 4 \\
 & V_{3,1} & V_{3,2} & V_{3,3} & V_{3,4} \\
4 & 4\ I_{3,1} & 4\ I_{3,2} & 2\ W_{3,3} & 2
\end{array}$$

The variables 'V' and 'I' for stages 1 and 2 are determined using equations (4.5a) and (4.6a) as demonstrated below:

$V_{1,2} = M_{2,1} - M_{1,2} = 1$, $I_{1,1} = 0$

$V_{1,3} = (V_{1,2} + M_{2,2}) - M_{1,3} = -2 = 0$, $I_{1,2} = 2$

$V_{2,2} = M_{3,1} - (M_{2,2} + I_{1,2}) = -1 = 0$, $I_{2,1} = 1$

$V_{2,3} = (V_{2,2} + M_{3,2}) - M_{2,3} = 3$, $I_{2,2} = 0$

$V_{3,2} = M_{4,1} - (M_{3,2} + I_{2,2}) = -1 = 0$, $I_{3,1} = 1$

$V_{3,3} = (V_{3,2} + M_{4,2}) - M_{3,3} = 1$, $I_{3,2} = 0$

The variables 'V' and 'W' for stage 3 are determined using equations (4.7a) and (4.8a) as demonstrated below:

$V_{1,4} = (M_{2,3} + V_{1,3}) - M_{1,4}) = -2 = 0$, $W_{1,3} = 2$

$V_{2,4} = (M_{2,3}+M_{3,3}+V_{1,3}+V_{2,3}) - (M_{1,4}+M_{2,4}+V_{1,4}) = 2$, $W_{2,3} = 0$

$V_{3,4}=(M_{2,3}+M_{3,3}+M_{4,3}+V_{1,3}+V_{2,3}+V_{3,3}) - (M_{1,4}+M_{2,4}+M_{3,4}+V_{1,4}+V_{2,4}) = -1=0$, $W_{3,3} = 1$

Using the results obtained above, the makespan is calculated using equation (4.4).

$$Makespan = \sum_{j=1}^{m} M_{1,j} + \sum_{i=2}^{n} M_{i,m} + \sum_{i=1}^{n-1} V_{i,m} = 33 \text{ hours}$$

Note that the result obtained is equal to the result obtained from the Gantt chart method as demonstrated in Example 4.12.

Number of Temporary Storages

Next, the number of temporary storages required is determined. From the calculation of variables above, it is recognized that the calculated values of variables $W_{1,3}$ *and* $W_{3,3}$are greater than 0. This indicates that there is requirement of two temporary storages in total i.e., after third stage as shown in Figure 4.16. The calculated values of these variables represent the duration of product intermediate inside these temporary storages. This is also in accordance with the Gantt chart shown in Figure 4.15 for the same batch process recipe.

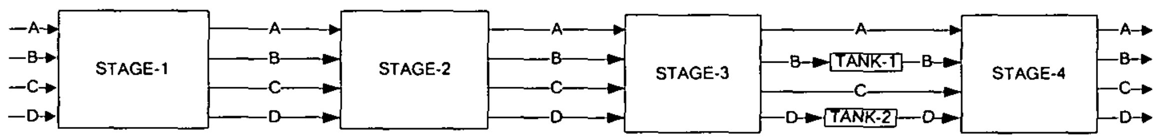


Figure 4.16: Temporary storage arrangement for production sequence ABCD for example 4.12

4.3 Summary

The makespan determination for specified production sequences of batch process does not offer a very challenging task when the problem size is small. This can be done easily by representing the batch process recipe on the Gantt chart. However, Gantt chart no longer remains efficient for the case when the problem size is large and requires extensive calculations especially for the batch process operated under different transfer policies. For this purpose, past researchers have developed various mathematical methods. These mathematical methods are built on a series of equations involving sets of variables and constraints used to solve the batch scheduling problem.

The developed method using matrix representation offers an alternative for makespan determination of a given batch process. The batch process recipes are represented in the form of matrix where the products are arranged according to the process in rows and stages are arranged in columns respectively. A procedure containing series of new mathematical equations derived from the understanding developed from Gantt chart based on different batch process recipe, is applied to the matrix to determine the corresponding makespan. Some other benefits of the developed method include the determination of idle time between stages, holding time inside the same stage and waiting time inside the temporary storage according to the transfer policy adopted. A summary of the mathematical formulations developed for makespan determination for various transfer policies is shown in Appendix B.

CHAPTER 5

EFFECT OF TRANSFER AND SETUP TIME

For the purpose of developing the proposed method for makespan determination earlier, the transfer and setup time were assumed negligible. However, for real world applications, the transfer and setup time can not be neglected. The transfer time refers to the time taken by product intermediate for its transfer from one stage to the other. The setup time refers to the time taken to setup a stage prior to it accepting subsequent product intermediate from the earlier stage. It is obvious that the makespan of a specific production sequence for a given batch process will increase with the inclusion of transfer and setup time. Moreover, many past studies recommended consideration to be given to sequence dependent transfer and setup time for batch process scheduling. This is due to the fact that the transfer and setup time would vary with different production sequence. Therefore, one could anticipate that the optimal production sequence(s) obtained using the proposed method in Chapter 5 may not necessarily be the same when the sequence dependent transfer and setup time are included.

An example of a batch process producing two products in three stages is used to illustrate the effect of introducing the transfer and setup time. The Gantt charts in Figures 5.1 and 5.2 show that the transfer and setup time can be adjusted along the process makespan using the information on the idle times between process stages. For example, in Figure 5.1, the idle time exists between the two products at stage 2 and 3. However, it is none at stage 1. The idle time, if exists and sufficient, can be used as transfer and setup time. The additional time for transfer and setup will only be required if the idle time, provided by the process itself, is not present or insufficient. The mathematical expressions developed for makespan determination for all transfer policies in Chapter 4 can be modified to include the time for transfer and setup between process stages.

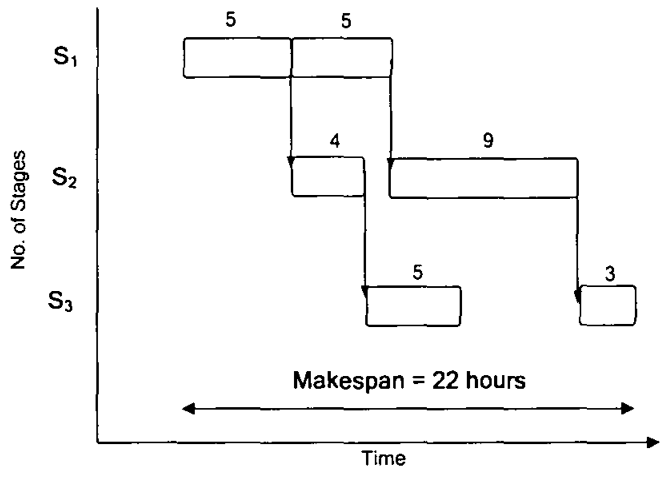


Figure 5.1: Gantt chart for a batch process without transfer and setup time

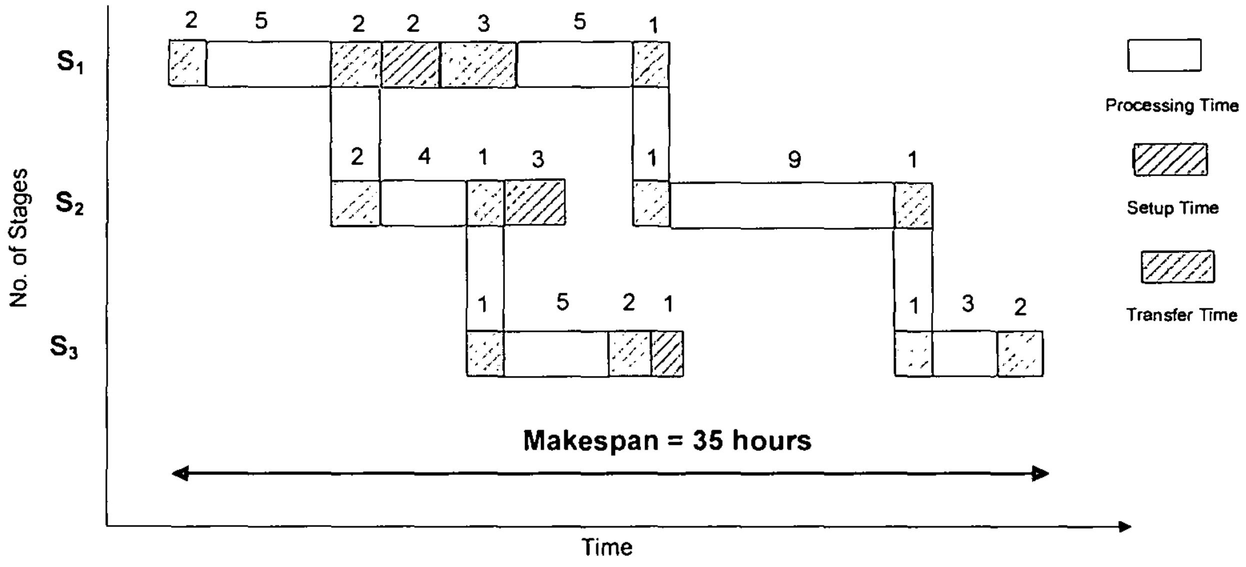


Figure 5.2: Gantt chart for a batch process with transfer and setup time

The preliminary information required are the transfer and setup time data for each product in a multiproduct batch process. For the purpose of explanation, the proposed procedure to include transfer and setup time with makespan calculation is discussed for each transfer policy separately. For illustration purpose, the calculation procedures for transfer and setup time are demonstrated using the production sequences from Chapter 4 for each transfer policy.

5.1 ZW transfer policy

The production sequence in example 4.3 i.e., ABC with process makespan of 66 hours, is used to illustrate the recalculation of makespan with transfer and setup time. For ease of calculation, the transfer and setup time data for the respective stages for all the products is represented in rows and columns as shown in Tables 5.1 and 5.2 below. The variables, T and U, are used to represent the transfer and setup time respectively in the forthcoming calculations.

Table 5.1
Stage transfer time for production sequence ABC for example 4.3

Products	Transfer time (hour) T				
		columns (j)			
	rows (i)	0	1	2	3
A	1	3	2	2	1
B	2	2	3	2	2
C	3	2	3	2	2

Table 5.2
Stage setup time for production sequence ABC for example 4.3

Products	Setup time (hour) U			
		columns (j)		
	rows (i)	1	2	3
A & B	1	1	3	2
B & C	2	4	1	2

Transfer Time

Earlier in Chapter 4, for the determination of makespan without transfer and setup time, the variables 'V' between process stages are calculated using the procedure of forward and reverse calculation. Here, the same procedure is repeated with the introduction of a variable 'T' to represent the transfer time. As such, the equations (4.1), (4.2) and (4.3) developed for the case of ZW transfer policy in Chapter 4 are re-formulated as follows:

$$V_{i,2} = (M_{i+1,1} + T_{i+1,0}) - (M_{i,2} + T_{i,2}) \qquad i = 1......n-1. \qquad (4.1a)$$

$$V_{i,j+1} = (V_{i,j} + M_{i+1,j} + T_{i+1,j-1}) - (M_{i,j+1} + T_{i,j+1}) \qquad j = 2........m-1,\ i = 1......n-1 \qquad (4.2a)$$

$$V_{i,j} = (V_{i,j+1} - (M_{i+1,j} + T_{i+1,j-1})) + (M_{i,j+1} + T_{i,j+1}) \qquad j = m-1........1,\ i = 1......n-1 \qquad (4.3a)$$

Similar to the earlier procedure, for any negative value of 'V', zero value will be taken instead. The process recipe of example 4.3 for the production sequence ABC from Chapter 4 will be used to illustrate the application of above equations as follows:

	1	2	3
1	10	20	5
	$V_{1,1}$	$V_{1,2}$	$V_{1,3}$
2	15	8	12
	$V_{2,1}$	$V_{2,2}$	$V_{2,3}$
3	20	7	9

The variables $V_{1,2}$ and $V_{2,2}$ are calculated using equation 4.1a, and, $V_{1,3}$ and $V_{2,3}$ are calculated using equation (4.2a) as follows:

$$V_{1,2} = (M_{2,1} + T_{2,0}) - (M_{1,2} + T_{1,2}) = -5 = 0$$

$$V_{2,2} = (M_{3,1} + T_{3,0}) - (M_{2,2} + T_{2,2}) = 12$$

$$V_{1,3} = (V_{1,2} + M_{2,2} + T_{2,1}) - (M_{1,3} + T_{1,3}) = 5$$

$$V_{2,3} = (V_{2,2} + M_{3,2} + T_{3,1}) - (M_{2,3} + T_{2,3}) = 8$$

Based on the calculated values of $V_{1,3}$ and $V_{2,3}$, the values of $V_{1,2}$ and $V_{2,2}$ are recalculated using equation (4.3a) to determine the values of $V_{1,1}$ and $V_{2,1}$ as demonstrated below:

$$V_{1,2} = (V_{1,3} - (M_{2,2} + T_{2,1})) + (M_{1,3} + T_{1,3}) = 0$$

$$V_{1,1} = (V_{1,2} - (M_{2,1} + T_{2,0})) + (M_{1,2} + T_{1,2}) = 8$$

$$V_{2,2} = (V_{2,3} - (M_{3,2} + T_{3,1})) + (M_{2,3} + T_{2,3}) = 8$$

$$V_{2,1} = (V_{2,2} - (M_{3,1} + T_{3,0}) + (M_{2,2} + T_{2,2}) = -4 = 0$$

The idle times calculated after addition of transfer times are shown in Table 5.3.

Table 5.3

Idle time calculated for production sequence ABC for example 4.3

Products	Idle time (hour)			Products	Idle time (hour)		
	$V_{1,1}$	$V_{1,2}$	$V_{1,3}$		$V_{2,1}$	$V_{2,2}$	$V_{2,3}$
A & B	8	0	5	B & C	0	8	8

Setup Time

The setup time calculation for each stage in a batch process could be done using the information on idle times between process stages. For example, from the information on idle times presented in Table 5.3 for the production sequence BAC, the setup time for each stage can be adjusted. The idle times given in Table 5.3 may or may not be sufficient to meet the criteria of required setup times shown in Table 5.2. Therefore, the additional time required over the idle times to meet the required setup times can be calculated by subtracting the idle times given in Table 5.3 from required setup times given in Table 5.2. The subtracted values are then analyzed for the maximum value. The maximum subtracted value will be added to the existing idle times between all stages of the respective products. For example, the idle times represented as variables, V, are subtracted from setup times, U, for the production sequence BAC. The results of the subtraction are shown using another variable, CT in Table 5.4.

$CT_{1,1}=U_{1,1}-V_{1,1}=-7$ $CT_{1,2}=U_{1,2}-V_{1,2}=3$ $CT_{1,3}=U_{1,3}-V_{1,3}=-3$

$CT_{2,1}=U_{2,1}-V_{2,1}=4$ $CT_{2,2}=U_{2,2}-V_{2,2}=-7$ $CT_{2,3}=U_{2,3}-V_{2,3}=-6$

Table 5.4

Subtracted idle time (Table 5.3) from stage setup time (Table 5.2)

Products.	CT (hour)			
		columns (j)		
	rows (i)	1	2	3
A & B	1	-7	3	-3
B & C	2	4	-7	-6

The negative value indicates that the idle time is sufficient for that stage to be used as required setup time while positive value indicates that additional time is required over the idle time to fulfill the setup time requirement as per Table 5.2.

The maximum values obtained are 3 and 4 between products A and B, and, products B and C respectively as shown in Table 5.4. These maximum values will be added to the corresponding idle times calculated in Table 5.3 and are shown in Figure 5.3 using a Gantt chart.

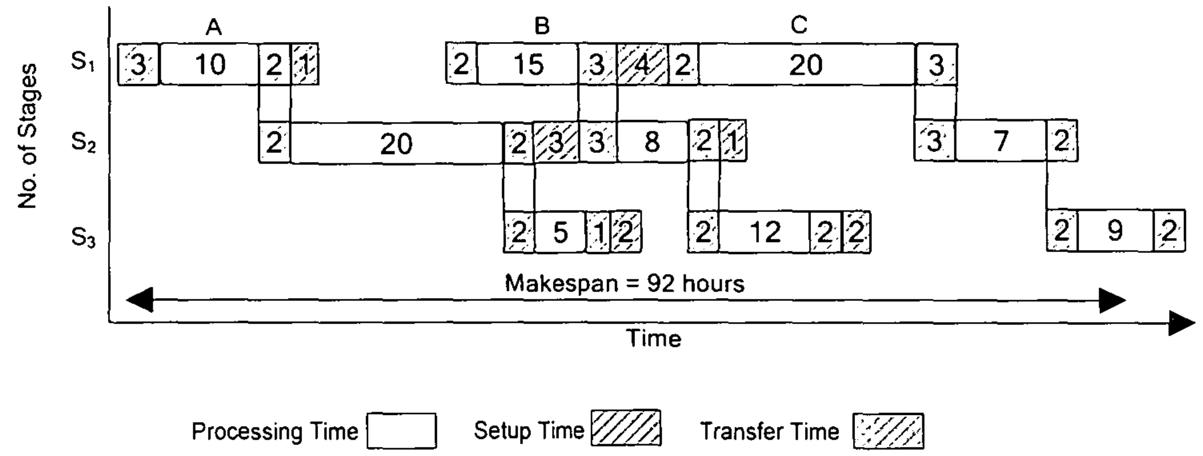


Figure 5.3: Gantt chart for production sequence ABC with transfer and setup time for example 4.3

The mathematical equation developed for makespan determination without transfer and setup time in Chapter 4 is reformulated to recalculate the makespan of the production sequence ABC with transfer and setup time as shown below:

$$Makespan = \sum_{j=1}^{3} M_{1,j} + \sum_{i=2}^{3} M_{i,3} + \sum_{i=1}^{2} H_{i,3} + \sum_{j=0}^{3} T_{1,j} + \sum_{i=2}^{3} (T_{i,m-1} + T_{i,m}) = 92 \qquad (4.4a)$$

$$Where \sum_{i=1}^{2} H_{i,3} = \sum_{i=1}^{2} V_{i,3} + \sum_{i=1}^{2} MAX(CT_{i,1}, CT_{i,2}, CT_{i,3}) = 20$$

For "n" number of products to be processed in "m" number of stages, the above equation for makespan determination with transfer and setup time is re-written in the form below:

$$Makespan = \sum_{j=1}^{m} M_{1,j} + \sum_{i=2}^{n} M_{i,m} + \sum_{i=1}^{n-1} H_{i,m} + \sum_{j=0}^{m} T_{1,j} + \sum_{i=2}^{n} (T_{i,m-1} + T_{i,m}) \qquad (4.4b)$$

$$Where \sum_{i=1}^{n-1} H_{i,m} = \sum_{i=1}^{n-1} V_{i,m} + \sum_{i=1}^{n-1} MAX(CT_{i,1}, CT_{i,2}, \ldots CT_{i,m})$$

The result of equation (4.4a) reiterates the fact stated earlier that the makespan of a batch process would increase with the inclusion of transfer and setup time.

Further test on the accuracy of the proposed method in determining the makespan value for larger problem size has been demonstrated on a batch process recipe with 10 products and 5 stages (Appendix A). The results for makespan value and inter-stage idle times have also been shown in agreement with a Gantt chart drawn for this purpose in Appendix A.

5.2 NIS/UW transfer policy

The production sequence in example 4.6 i.e., ABCD with process makespan of 40 hours, is used to illustrate the recalculation of makespan with transfer and setup time for the given batch process operated under NIS/UW. For ease of calculation, the transfer and setup time data for the respective products is arranged in rows and columns as shown in Tables 5.5

and 5.6 below. The variables, T and U, are used to represent the transfer and setup time respectively in the forthcoming calculations.

Table 5.5
Stage transfer time for production sequence ABCD for example 4.6

Products	Transfer time (hour) T				
		columns (j)			
	rows (i)	0	1	2	3
A	1	2	3	2	2
B	2	2	1	1	2
C	3	3	2	2	1
D	4	2	3	2	2

Table 5.6
Stage setup time for production sequence ABCD for example 4.6

Products	Setup time (hour) U			
		columns (j)		
	rows (i)	1	2	3
A & B	1	2	3	1
B & C	2	1	2	3
C & D	3	5	5	3

Transfer and Setup Time

For the determination of makespan in Chapter 4, the variables representing idle time, V, between process stages and holding time, I, of product intermediate inside the same stages were calculated without the inclusion of the transfer and setup time. Here, the same procedure is repeated with the introduction of a variable, T, to represent the transfer time from Table 5.5 above. As such, the equations (4.5) and (4.6) developed for the case of NIS/UW in Chapter 4, are re-formulated as follows:

$$V_{1,j+1} = (V_{1,j} + M_{2,j} + T_{2,j-1}) - (M_{1,j+1} + T_{1,j+1}) \qquad j = 1........m-1 \qquad (4.5b)$$

$$V_{i,j+1} = (V_{i,j} + M_{i+1,j} + T_{i+1,j-1}) - (M_{i,j+1} + T_{i,j+1} + I_{i-1,j+1}) \quad j = 1........m-1, \quad i = 2.......n\text{-}1 \quad (4.6b)$$

$$V_{i,1} = 0, \quad I_{i,m} = 0 \qquad i = 1.......n\text{-}1$$

As explained earlier in Chapter 4,if the calculated value of $V_{i,j+1}$ is negative, a zero value is assigned instead. However, the calculated value sign will be changed to positive and assigned to $I_{i,j}$ i.e., holding time in the earlier stage. Otherwise, $I_{i,j}$ will be zero.

For illustration purpose, the production sequence i.e., ABCD from example 4.7 is arranged to form a matrix with the introduction of variables, V and I as shown below.

$$\begin{array}{cccc}
 & 1 & 2 & 3 \\
1 & 3.5 & 4.3 & 8.7 \\
 & V_{1,1} & V_{1,2} & V_{1,3} \\
2 & 4.0 \;\; I_{1,1} & 5.5 \;\; I_{1,2} & 3.5 \\
 & V_{2,1} & V_{2,2} & V_{2,3} \\
3 & 3.5 \;\; I_{2,1} & 7.5 \;\; I_{2,2} & 6.0 \\
 & V_{3,1} & V_{3,2} & V_{3,3} \\
4 & 12.0 \; I_{3,1} & 3.5 \;\; I_{3,2} & 8.0
\end{array}$$

As concluded earlier in Chapter 4, the variables between the first stages for all products i.e., $V_{1,1}$, $V_{2,1}$ and $V_{3,1}$ will be zero. This is in accordance with the transfer policy adopted i.e., NIS/UW.

Next, a comparison will be made between the variables, V, and required setup times, U, given in Table 5.6 for the first stage of all the products i.e., $V_{1,1}$ and $U_{1,1}$, $V_{2,1}$ and $U_{2,1}$, $V_{3,1}$ and $U_{3,1}$. The highest value from the comparison made is selected. For calculation purpose, this highest value is represented using another variable, H. As such, the above procedure will produce values for $H_{1,1}$, $H_{2,1}$ and $H_{3,1}$ respectively. In fact, the values of $H_{1,1}$, $H_{2,1}$ and $H_{3,1}$ represent the time that would meet the criteria of required setup time given in Table 5.6 as shown below.

$V_{1,1} = 0$	$U_{1,1} = 2$	$H_{1,1}= 2$
$V_{2,1} = 0$	$U_{2,1} = 1$	$H_{2,1}= 1$
$V_{3,1} = 0$	$U_{3,1} = 5$	$H_{3,1}= 5$

The variable $V_{1,2}$ for second stage between the first two products is determined using equation (4.5b). However, the variable $V_{1,1}$ in equation (4.5b) will be replaced with variable $H_{1,1}$ due to the comparison made above. Next, the calculated value for variable $V_{1,2}$ is compared with the required setup time $U_{1,2}$ given in Table 5.6. The highest from the comparison made is selected and assigned as variable $H_{1,2}$ as shown below.

$V_{1,2} = (H_{1,1}+M_{2,1}+T_{2,0}) - (M_{1,2}+T_{1,2}) = 1.7,$ $U_{1,2}=3,$ $H_{1,2}=3$

The calculation of holding time, $I_{1,1}$, in the earlier stage could be done using the value of $H_{1,2}$ calculated above. As such, the following equation (5.1) is formulated.

$$I_{1,j} = (H_{1,j+1} + M_{1,j+1} + T_{1,j+1}) - (H_{1,j} + M_{2,j} + T_{2,j-1}) \qquad j = 1 \ldots\ldots\ldots\ldots m-1 \qquad (5.1)$$

$I_{1,1} = (H_{1,2} + M_{1,2}+T_{1,2}) - (H_{1,1}+M_{2,1}+T_{2,0}) = 1.3$

The same procedure could be repeated for calculating the variable for the third stage between the first two products i.e., $V_{1,3}$, using equation (4.5b). Again, this is done by replacing $V_{1,2}$ with $H_{1,2}$ in equation (4.5b). The calculated value of $V_{1,3}$ is compared with the required setup time $U_{1,3}$ given in Table 5.6. Again, the highest value from the comparison made is selected and assigned to variable $H_{1,3}$.

$V_{1,3} = (H_{1,2} + M_{2,2}+T_{2,1}) - (M_{1,3}+T_{1,3}) = -1.2=0,$ $U_{1,3}=1,$ $H_{1,3}=1$

The calculation of holding time, $I_{1,2}$, using equation (5.1) , in the earlier stage could be done using the value of $H_{1,3}$ calculated above.

$I_{1,2} = (H_{1,3} + M_{1,3}+T_{1,3}) - (H_{1,2} + M_{2,2}+T_{2,1}) = 2.2$

The variables between stages of the succeeding products are determined using equation (4.6b). Similar to the comparison procedure adopted earlier, the variable, V, is replaced with corresponding variable, H. The holding time, I, in the earlier stage is determined using equation (5.2) below.

$$I_{i,j} = (H_{i,j+1} + M_{i,j+1} + T_{i,j+1} + I_{i-1,j+1}) - (H_{i,j} + M_{i+1,j} + T_{i+1,j-1})$$
$$j = 1 \ldots\ldots m-1, \quad i = 2 \ldots\ldots n-1 \qquad (5.2)$$

$$I_{i,m} = 0 \qquad i = 1 \ldots\ldots\ldots n-1$$

$V_{2,2} = (H_{2,1} + M_{3,1}+T_{3,0}) - (M_{2,2} +T_{2,2}+ I_{1,2}) = -1.2=0$

$U_{2,2}=2,\ H_{2,2}=2$

$I_{2,1} = (H_{2,2} + M_{2,2} +T_{2,2}+ I_{1,2}) - (H_{2,1} + M_{3,1}+T_{3,0}) = 3.2$

$V_{2,3} = (H_{2,2} + M_{3,2}+T_{3,1}) - (M_{2,3} +T_{2,3}+I_{1,3})= 6$

$U_{2,3}=3,\ H_{2,3}=6$

$I_{2,2} = (H_{2,3} + M_{2,3} +T_{2,3}+ I_{1,3}) - (H_{2,2} + M_{3,2}+T_{3,1}) = 0$

$V_{3,2} = (H_{3,1} + M_{4,1}+T_{4,0}) - (M_{3,2} +T_{3,2}+ I_{2,2}) = 9.5$

$U_{3,2}=5,\ H_{3,2}=9.5$

$I_{3,1} = (H_{3,2} + M_{3,2} +T_{3,2}+ I_{2,2}) - (H_{3,1} + M_{4,1}+T_{4,0}) = 0$

$V_{3,3} = (H_{3,2} + M_{4,2}+T_{4,1}) - (M_{3,3} +T_{3,3}+I_{2,3}) = 9$

$U_{3,3}=3,\ H_{3,3}= 9$

$I_{3,2} = (H_{3,3} + M_{3,3} +T_{3,3}+ I_{2,3}) - (H_{3,2} + M_{4,2}+T_{4,1}) = 0$

Finally, the mathematical equation (4.4) developed for makespan determination without transfer and setup time in Chapter 4 is extended to recalculate the makespan of the production sequence ABCD with transfer and setup time as shown below. Based on the procedure used to include setup time earlier, all of the variables, V, in equation (4.4) will be replaced by the variable, H.

$$Makespan = \sum_{j=1}^{m} M_{1,j} + \sum_{i=2}^{n} M_{i,m} + \sum_{i=1}^{n-1} H_{i,m} + \sum_{j=0}^{m} T_{1,j} + \sum_{i=2}^{n} (T_{i,m-1} + T_{i,m}) = 69 \text{ hours} \qquad (4.4c)$$

The makespan obtained with the transfer and setup time for production sequence ABCD of the given batch process recipe with no intermediate storage transfer policy is illustrated using Gantt chart in Figure 5.4.

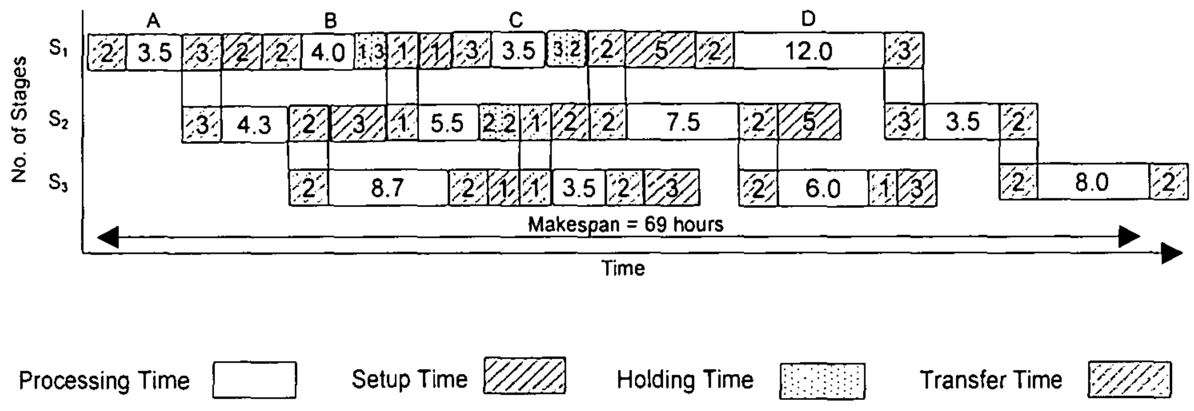


Figure 5.4: Gantt chart for production sequence ABCD with transfer and setup time for example 4.6

The result of equation (4.4c) validates the conclusion made earlier in the case of ZW transfer policy i.e., the makespan of a batch process would increase with the inclusion of transfer and setup time.

Similar to the ZW transfer policy, the application of proposed method for NIS/UW in determining the makespan as well as inter-stage idle times and holding times for the same problem size (10 products and 5 stages) has been demonstrated in Appendix A. The corresponding Gantt chart is also exhibited to verify the results.

5.3 UIS/UW transfer policy

The production sequence ABCD from example 4.8 with process makespan of 29 hours, is used to illustrate the recalculation of makespan with transfer and setup time for the case of batch process operated under UIS/UW. For ease of calculation, the transfer and setup time data for the respective products are represented in rows and columns as shown in Tables 5.7 and 5.8 below. The variables, T and U, are used to represent the transfer and setup time respectively in the forthcoming calculations. The setup time for temporary storage is assumed negligible because in UIS/UW, the temporary storage is always made available (Kim et al. 1996).

Table 5.7
Stage transfer time for production sequence ABCD for example 4.8

Products	rows (i)	Transfer time (hour) T columns (j) 0	1	2	3
A	1	2	2	2	2
B	2	1	2	3	2
C	3	2	3	3	2
D	4	1	2	1	1

Table 5.8
Stage setup time for production sequence ABCD for example 4.8

Products	rows (i)	Setup time (hour) U columns (j) 1	2	3
A & B	1	4	3	4
B & C	2	3	1	2
C & D	3	3	2	3

Transfer and Setup Time

For the determination of makespan in Chapter 4, the variables representing idle time, V, between process stages and waiting time, W, of product intermediate inside the temporary storage were calculated without including the transfer and setup time. Here, the same procedure is repeated with the introduction of a variable, T, to represent the transfer time from Table 5.7 above. For this purpose, the production sequence ABCD from example 4.8 is arranged to form a matrix as shown below. Firstly, the transfer times are added to the processing times for all the products in all stages as represented in the following matrix.

$$\begin{array}{cccc} & 1 & 2 & 3 \\ 1 & T_{1,0}+5+T_{1,1} & T_{1,2}+8 & T_{1,3}+6 \\ 2 & T_{2,0}+6 & T_{2,1}+5 & T_{2,2}+2 \\ 3 & T_{3,0}+3 & T_{3,1}+5 & T_{3,2}+3 \\ 4 & T_{4,0}+3 & T_{4,1}+4 & T_{4,2}+2 \end{array}$$

Next, the variables, V, used to represent idle times between process stages and waiting time, W, of product intermediate inside the temporary storage, are introduced in the resulting matrix after the addition of transfer times to the processing times above.

$$\begin{array}{cccc}
 & 1 & 2 & 3 \\
1 & 9 & 10 & 8 \\
 & V_{1,1} & V_{1,2} & V_{1,3} \\
2 & 7\ W_{1,1} & 7\ W_{1,2} & 5 \\
 & V_{2,1} & V_{2,2} & V_{2,3} \\
3 & 5\ W_{2,1} & 8\ W_{2,2} & 6 \\
 & V_{3,1} & V_{3,2} & V_{3,3} \\
4 & 4\ W_{3,1} & 6\ W_{3,2} & 3
\end{array}$$

As concluded earlier in Chapter 4, the variables between the first stages for all products i.e., $V_{1,1}$, $V_{2,1}$ and $V_{3,1}$ will be zero. This is in accordance with the transfer policy adopted i.e., UIS.

Next, similar to NIS/UW, a comparison will be made between the variables, V, and required setup times, U, given in Table 5.8 for the first stage of all products. The highest values from the comparison made are selected and assigned as variable, H. As such, the comparison procedure will produce respective values of $H_{1,1}$, $H_{2,1}$ and $H_{3,1}$ as shown below.

$$V_{1,1} = 0 \qquad U_{1,1} = 4 \qquad H_{1,1} = 4$$

$$V_{2,1} = 0 \qquad U_{2,1} = 3 \qquad H_{2,1} = 3$$

$$V_{3,1} = 0 \qquad U_{3,1} = 3 \qquad H_{3,1} = 3$$

The variable,$V_{1,2}$, for second stage between the first two products is determined using equation (4.7). However, in equation (4.7), the variable "$V_{1,1}$" will be replaced with variable "$H_{1,1}$" as shown above. The calculated value for variable, $V_{1,2}$ is compared with the required setup time, $U_{1,2}$, given in Table 5.8. The highest value from the comparison made is selected and assigned as variable, $H_{1,2}$.

$V_{1,2} = (M_{2,1}+H_{1,1}) - M_{1,2} = 1$, $U_{1,2}=3$, $H_{1,2}= 3$

The waiting time, $W_{1,1}$, in the temporary storage is calculated using equation (5.3) below.

$$\begin{bmatrix} if\ H_{1,j+1} > V_{1,j+1},\quad W_{1,j} = (H_{1,j+1} + M_{1,j+1}) - (H_{1,j} + M_{2,j} + T_{2,j})\ \ else\ \ W_{1,j} = 0 \\ if\ (H_{1,j+1} > V_{1,j+1}\ \ and\ \ W_{1,j} \le 0),\quad H_{1,j+1} = H_{1,j+1} + (-W_{1,j})\ \ and\ \ W_{1,j} = 0\ (pass) \end{bmatrix}$$
$$j = 1 \ldots\ldots\ldots m\text{-}1 \quad (5.3)$$

$W_{1,1} = (H_{1,2} + M_{1,2}) - (H_{1,1}+ M_{2,1}+T_{2,1}) = 0$ (pass)

The same procedure could be repeated for calculating the variable for the third stage between the first two products i.e., $V_{1,3}$ using equation (4.7). Again, in equation (4.7), the variable $V_{1,2}$ is replaced with variable $H_{1,2}$. The calculated value of variable, $V_{1,3}$, is compared with the required setup time $U_{1,3}$ given in Table 5.8. The highest value from the comparison made is selected and assigned as variable, $H_{1,3}$.

$V_{1,3} = (M_{2,2} +H_{1,2}) - M_{1,3} = 2$ $U_{1,3}=4$, $H_{1,3}= 4$

The waiting time, $W_{1,2}$, in the temporary storage is calculated using equation (5.3) above.

$W_{1,2} = (H_{1,3} + M_{1,3}) - (H_{1,2}+ M_{2,2}+T_{2,2}) = -1 = 0$ (pass)

$H_{1,3} = 1+4 = 5$

The variables between stages of the succeeding products are determined using equation (4.8b). The variable, V, is replaced with variable, H. The waiting time, W, is determined using equation (5.4) below.

$$V_{i,j} = (\sum_{k=2}^{i+1} M_{k,j-1} + \sum_{k=2}^{i} T_{k,j-1} + \sum_{k=1}^{i} H_{k,j-1}) - (\sum_{k=1}^{i} M_{k,j} + \sum_{k=2}^{i} T_{k,j} + \sum_{k=1}^{i-1} H_{k,j})$$

$$j = 2.......m, \quad i = 2........n-1 \qquad (4.8b)$$

$$\left[\begin{array}{l} if\ H_{i,j} > V_{i,j},\ W_{i,j-1} = (\sum_{k=1}^{i} M_{k,j} + \sum_{k=2}^{i} T_{k,j} + \sum_{k=1}^{i} H_{k,j}) - (\sum_{k=2}^{i+1} M_{k,j-1} + \sum_{k=2}^{i+1} T_{k,j-1} + \sum_{k=1}^{i} H_{k,j-1}) \\ \\ else\ W_{i,j-1}=0 \\ \\ if\ (H_{i,j} > V_{i,j}\ \ and\ \ W_{i,j-1} \le 0), \quad H_{i,j} = H_{i,j} + (-W_{i,j-1})\ \ and\ \ W_{i,j-1} = 0\ (pass) \end{array}\right]$$

$$j = 2.......m, \quad i = 2........n-1 \quad (5.4)$$

For any negative or zero value of, W, obtained from equations (5.3) and (5.4), it will be added to the corresponding variable, H, as demonstrated below for the case of $W_{2,2}$. The negative or zero value of, W, represents that instead of waiting, the product intermediate will only pass through the temporary storage before it is transferred to the next stage. This is because the time delay in the availability of the next stage becomes equal or less than the transfer time of the product intermediate to and from the temporary storage as shown in Figure 6.6 for the product intermediate of third product after second stage. For the purpose of calculation in such cases, the, W, will be assigned a zero value and the word "pass" will be used to demonstrate it.

$V_{2,2} = (M_{2,1}+M_{3,1}+T_{2,1}+H_{1,1}+H_{2,1}) - (M_{1,2}+M_{2,2}+T_{2,2}+H_{1,2}) = -2 = 0$

$U_{2,2}=1, \quad H_{2,2}=1$

$W_{2,1} = (M_{1,2}+M_{2,2}+T_{2,2}+H_{1,2}+H_{2,2}) - (M_{2,1}+M_{3,1}+T_{2,1}+T_{3,1}+H_{1,1}+H_{2,1}) = 0$ (pass)

$V_{2,3} = (M_{2,2}+M_{3,2}+T_{2,2}+H_{1,2}+H_{2,2}) - (M_{1,3}+M_{2,3}+T_{2,3}+H_{1,3}) = 2$

$U_{2,3}=2, \quad H_{2,3}=2$

$W_{2,2} = 0$

$V_{3,2}= (M_{2,1}+M_{3,1}+M_{4,1}+T_{2,1}+T_{3,1}+H_{1,1}+H_{2,1}+H_{3,1}) - (M_{1,2}+M_{2,2}+M_{3,2}+T_{2,2}+T_{3,2}+H_{1,2}+H_{2,2})$
$= -4 = 0$

$U_{3,2}=2$, $H_{3,2}=2$

$W_{3,1}=(M_{1,2}+M_{2,2}+M_{3,2}+T_{2,2}+T_{3,2}+H_{1,2}+H_{2,2}+H_{3,2})-(M_{2,1}+M_{3,1}+M_{4,1}+T_{2,1}+T_{3,1}+$
$T_{4,1}+H_{1,1}+H_{2,1}+H_{3,1}) = 4$

$V_{3,3}= (M_{2,2}+M_{3,2}+M_{4,2}+T_{2,2}+T_{3,2}+H_{1,2}+H_{2,2}+H_{3,2}) - (M_{1,3}+M_{2,3}+M_{3,3}+T_{2,3}+T_{3,3}+H_{1,3}+H_{2,3})$
$= 3$

$U_{3,3}=3$, $H_{3,3}=3$, $W_{3,2} = 0$

Finally, the mathematical equation (4.4) developed for makespan determination without transfer and setup time in Chapter 4 is reformulated to calculate the makespan of the production sequence ABCD with transfer and setup time as shown below. Based on the procedure used to include setup time earlier, all of the variables, V, in equation (4.4) will be replaced by the variable, H.

$$Makespan = \sum_{j=1}^{m} M_{1,j} + \sum_{i=2}^{n} M_{i,m} + \sum_{i=1}^{n-1} H_{i,m} + \sum_{i=2}^{n} T_{i,m} \quad = 56 \text{ hours} \qquad (4.4d)$$

The illustration of transfer and setup time location between the stages for all the products for the production sequence ABCD of the given batch process recipe with UIS/UW transfer policy is done using Gantt chart in Figure 5.5.

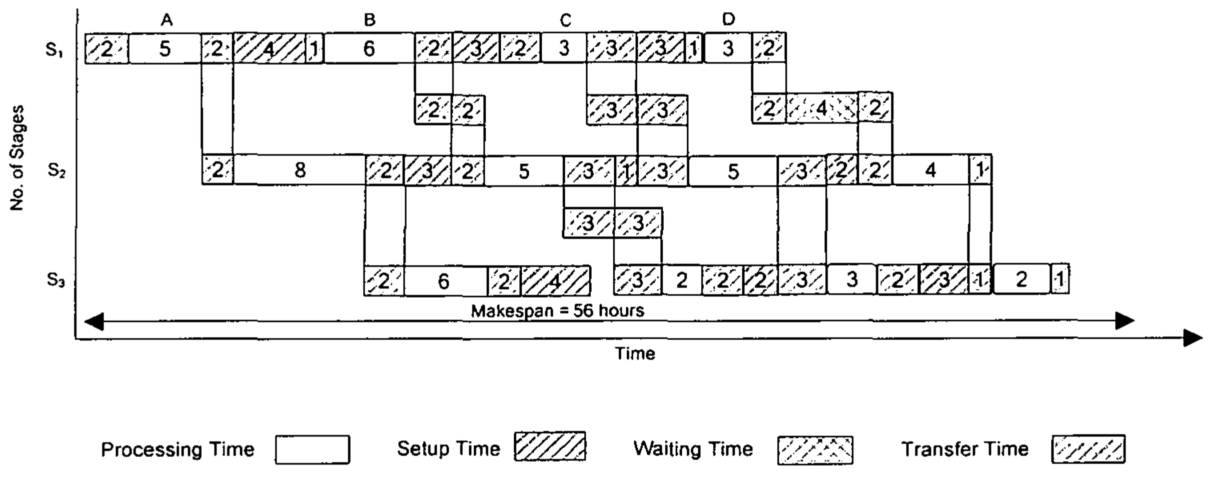


Figure 5.5: Gantt chart for production sequence ABCD with transfer and setup time for example 4.8

Number of Storage Tanks

Next, the number of temporary storage required is determined. Since the values of $W_{1,1}$ and $W_{2,1}$ are zero and $W_{3,1}$ is greater than 0, they indicate the need for three temporary storage after the first stage. The zero value obtained in case of $W_{1,1}$ and $W_{2,1}$ shows that the product has only to pass through the storage tanks before transferring to the next stage. Also, negative value of $W_{1,2}$ shows that there is a need of another temporary storage after the second stage from which product has only to pass instead of waiting inside it. The batch process arrangement in this scenario is illustrated in Figure 5.6.

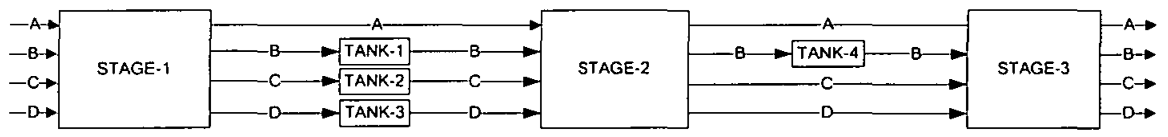


Figure 5.6.: Temporary storage arrangement for production sequence ABCD for example 4.8

The result of equation (4.4d) again validates the conclusion made earlier in the case of ZW and NIS/UW transfer policies i.e., the makespan of a batch process would increase with the inclusion of transfer and setup time.

5.4 FIS/UW transfer policy

The production sequence ABCD from example 4.10 with process makespan of 33 hours, is used to illustrate the recalculation of makespan with transfer and setup time for the case of FIS/UW. For ease of calculation, the transfer and setup time data for respective stages of all the products is represented in rows and columns as shown in Tables 5.9 and 5.10 below. The variables, T and U, are used to represent the transfer and setup time respectively in the forthcoming calculations.

In FIS/UW, the setup time for temporary storage will also be included because same storages are used for the succeeding products (Jung et al. 1996). For this purpose, the temporary storage setup time, G, for respective products are shown in Table 5.11.

Table 5.9
Stage transfer time for production sequence ABCD for example 4.10

Products		Transfer time (hour) T			
		columns (j)			
	rows (i)	0	1	2	3
A	1	2	3	2	1
B	2	2	3	2	2
C	3	3	2	2	2
D	4	2	1	1	3

Table 5.10
Stage setup time for production sequence ABCD for example 4.10

Products		Setup time (hour) U		
		columns (j)		
	rows (i)	1	2	3
A & B	1	1	2	3
B & C	2	3	2	2
C & D	3	2	2	3

Table 5.11

Storage setup time for production sequence ABCD for example 4.10

Products	Setup time (hour) G			
		columns (j)		
	rows (i)	1	2	3
A	1	2	2	2
B	2	3	4	6
C	3	2	3	2
D	4	2	3	2

Transfer and Setup Time

For the determination of makespan in Chapter 4, the variables representing idle time 'V' waiting time 'W' and holding time 'I' were calculated without including the transfer and setup time. Here, all these variables will be recalculated after inclusion of the transfer and setup time in the given batch process recipe. For this purpose, the batch process recipe from example 4.10 for production sequence ABCD is arranged to form a matrix as shown below. Firstly, the transfer times are added to the processing times of all the products in all stages as represented by variable, T, in the following matrix.

$$\begin{array}{cccc} & 1 & 2 & 3 \\ 1 & T_{1,0}+4+T_{1,1} & T_{1,2}+10 & T_{1,3}+5 \\ 2 & T_{2,0}+12 & T_{2,1}+4 & T_{2,2}+7 \\ 3 & T_{3,0}+3 & T_{3,1}+3 & T_{3,2}+4 \\ 4 & T_{4,0}+2 & T_{4,1}+2 & T_{4,2}+2 \end{array}$$

Next, the variables, representing idle time, V, waiting time, W, and holding time, I, are introduced in the matrix resulted after the addition of transfer times to the processing times above.

$$\begin{array}{cccc}
 & 1 & 2 & 3 \\
1 & 9 & 12 & 6 \\
 & V_{1,1} & V_{1,2} & V_{1,3} \\
2 & 14\ W_{1,1}I_{1,1} & 7\ W_{1,2}I_{1,2} & 9 \\
 & V_{2,1} & V_{2,2} & V_{2,3} \\
3 & 6\ W_{2,1},I_{2,1} & 5\ W_{2,2}I_{2,2} & 6 \\
 & V_{3,1} & V_{3,2} & V_{3,3} \\
4 & 4\ W_{3,1}I_{3,1} & 3\ W_{3,2}I_{3,2} & 3
\end{array}$$

As stated earlier in Chapter 4, the variables between the first stages for all products i.e., $V_{1,1}$, $V_{2,1}$ and $V_{3,1}$ will be zero. This is in accordance with the transfer policy adopted i.e., FIS/UW.

Next, similar to NIS/UW and UIS/UW, a comparison will be made between the variables, V, and required setup times,U, given in Table 5.10 for the first stage of all products. The highest value is selected from the comparison made and its value is assigned to variable, H. As such, the comparison procedure will produce respective values of $H_{1,1}$, $H_{2,1}$ and $H_{3,1}$ as shown below.

$V_{1,1} = 0$	$U_{1,1} = 1$	$H_{1,1}= 1$
$V_{2,1} = 0$	$U_{2,1} = 3$	$H_{2,1}= 3$
$V_{3,1} = 0$	$U_{3,1} = 2$	$H_{3,1}= 2$

The variable, $V_{1,2}$, is determined for the second stage between the first two products using equation (4.9). However, in equation (4.9), the variable, $V_{1,1}$, will be replaced with variable, $H_{1,1}$, as shown above. A comparison is made between the calculated value of variable, $V_{1,2}$, and the required setup time, $U_{1,2}$, given in Table 5.10. The highest value is selected from the comparison and assigned to variable, $H_{1,2}$.

$V_{1,2} = (M_{2,1}+H_{1,1}) - M_{1,2} = 3$ $\qquad U_{1,2}=2$, $\qquad H_{1,2}= 3$

The waiting time, $W_{1,1}$, in the temporary storage is then calculated using equation (5.5) below.

$$\begin{bmatrix} if\ H_{1,j+1} > V_{1,j+1},\ W_{1,j} = (H_{1,j+1} + M_{1,j+1}) - (H_{1,j} + M_{2,j} + T_{2,j})\ else\ W_{1,j} = 0 \\ if\ (H_{1,j+1} > V_{1,j+1}\ and\ W_{1,j} \le 0),\quad H_{1,j+1} = H_{1,j+1} + (-W_{1,j})\ and\ W_{1,j} = 0\ (pass) \end{bmatrix}$$
$$j = 1.........m-1 \qquad (5.5)$$

$W_{1,1} = 0$

The same procedure could be repeated for calculating the variable for the third stage between the first two products i.e., $V_{1,3}$ using equation (4.9). Again, in equation (4.9), variable, $V_{1,2}$, is replaced with variable $H_{1,2}$. The calculated variable, $V_{1,3}$, is compared with the required setup time $U_{1,3}$ given in Table 5.10. Again, the highest value from the comparison made is selected and assigned as variable, $H_{1,3}$.

$V_{1,3} = (M_{2,2} + H_{1,2}) - M_{1,3} = 4$ $\qquad U_{1,3}=3$, $\qquad H_{1,3}= 4$

Next, the waiting time, $W_{1,2}$, in the temporary storage is determined using equation (5.5) above.

$W_{1,2} = 0$

The variables between stages of the succeeding products are determined using equation (4.10a). The variable, V, is replaced with corresponding variable, H, according to the comparison procedure explained earlier. The waiting time, W, is determined using equation (5.6) below.

$$\left[\begin{array}{l} if\ G_{i,j-1} \le (M_{i+1,j-1} + H_{i,j-1}) - (W_{i-1,j-1} + T_{i,j-1}),\ G_{i,j-1} = 0 \\ \\ if\ (M_{i+1,j-1} + H_{i,j-1}) \le (W_{i-1,j-1} + T_{i,j-1} + G_{i,j-1}),\ I_{i,j-1} = (W_{i-1,j-1} + T_{i,j-1} + G_{i,j-1}) - (M_{i+1,j-1} + H_{i,j-1}) \\ else\ I_{i,j-1} = 0 \\ \\ V_{i,j} = (\sum_{k=2}^{i+1} M_{k,j-1} + \sum_{k=2}^{i} T_{k,j-1} + \sum_{k=1}^{i} H_{k,j-1} + \sum_{k=1}^{i} I_{k,j-1}) - (\sum_{k=1}^{i} M_{k,j} + \sum_{k=2}^{i} T_{k,j} + \sum_{k=1}^{i-1} H_{k,j} + \sum_{k=1}^{i-1} I_{k,j}) \end{array}\right]$$

$$j = 2......m,\ i = 2......n-1 \qquad (4.10a)$$

$$\left[\begin{array}{l} if\ H_{i,j} > V_{i,j},\ W_{i,j-1} = (\sum_{k=1}^{i} M_{k,j} + \sum_{k=2}^{i} T_{k,j} + \sum_{k=1}^{i} H_{k,j} + \sum_{k=1}^{i-1} I_{k,j}) - (\sum_{k=2}^{i+1} M_{k,j-1} + \sum_{k=2}^{i+1} T_{k,j-1} + \sum_{k=1}^{i} H_{k,j-1} + \sum_{k=1}^{i} I_{k,j-1}) \\ else\ W_{i,j-1} = 0 \\ \\ if\ (H_{i,j} > V_{i,j}\ and\ W_{i,j-1} \le 0),\quad H_{i,j} = H_{i,j} + (-W_{i,j-1})\ and\ W_{i,j-1} = 0\ (pass) \end{array}\right]$$

$$j = 2......m,\ i = 2......n-1 \qquad (5.6)$$

$$I_{1,j} = 0,\quad I_{i,m} = 0 \qquad\qquad i = 1.........n-1,\ j = 1........m$$

It is important to note that the setup time for temporary storage, G, would be considered into the above calculations after the processing of product intermediates of second product as shown in equation (4.10a). This is because no temporary storage is required after the processing of product intermediates of first product due to availability of all process stages.

As explained earlier in the case of UIS/UW, a negative or zero value of W obtained from equations (5.5) and (5.6) above will be added to H. The negative or zero value of W represents that instead of waiting, the product intermediate will only pass through the temporary storage before it is transferred to the next stage.

$I_{2,1} = 0$

$V_{2,2}= (M_{2,1}+M_{3,1}+T_{2,1}+H_{1,1}+H_{2,1}+I_{1,1}+I_{2,1}) - (M_{1,2}+M_{2,2}+T_{2,2}+H_{1,2}+I_{1,2}) = 3$

$U_{2,2}= 2,\quad H_{2,2} = 3,\quad W_{2,1} = 0$

$I_{2,2} = 0$

$V_{2,3}= (M_{2,2}+M_{3,2}+T_{2,2}+H_{1,2}+H_{2,2}+I_{1,2}+I_{2,2}) - (M_{1,3}+M_{2,3}+T_{2,3}+H_{1,3}+I_{1,3}) = -1=0$

$U_{2,3}= 2,\quad H_{2,3} = 2,\quad W_{2,2} = 1$

$W_{2,2} = (M_{1,3}+M_{2,3}+T_{2,3}+H_{1,3}+H_{2,3}+I_{1,3}) - (M_{2,2}+M_{3,2}+T_{2,2}+T_{3,2}+H_{1,2}+H_{2,2}+I_{1,2}+I_{2,2}) = 1$

$I_{3,1} = 0$

$V_{3,2} = (M_{2,1}+M_{3,1}+M_{4,1}+T_{2,1}+T_{3,1}+H_{1,1}+H_{2,1}+H_{3,1}+I_{1,1}+I_{2,1}+I_{3,1}) -$
$(M_{1,2}+M_{2,2}+M_{3,2}+T_{2,3}+T_{3,3}+H_{1,2}+H_{2,2}+I_{1,2}+I_{2,2}) = 1$

$U_{3,2}= 2,\quad H_{3,2} = 2$

$W_{3,1}$ = ($M_{1,2}$+$M_{2,2}$+$M_{3,2}$+$T_{2,2}$+$T_{3,2}$+$H_{1,2}$+$H_{2,2}$+$H_{3,2}$+$I_{1,2}$+$I_{2,2}$) –
($M_{2,1}$+$M_{3,1}$+$M_{4,1}$+ $T_{2,1}$+$T_{3,1}$+$T_{4,1}$+ $H_{1,1}$+$H_{2,1}$+$H_{3,1}$+$I_{1,1}$+$I_{2,1}$+$I_{3,1}$) = 0 (pass)

$I_{3,2}$ = ($W_{2,2}$+$T_{3,2}$+$G_{3,2}$) – ($M_{4,2}$+$H_{3,2}$) = 1

$V_{3,3}$ = ($M_{2,2}$+$M_{3,2}$+$M_{4,2}$+$T_{2,2}$+$T_{3,2}$+$H_{1,2}$+$H_{2,2}$+$H_{3,2}$+$I_{1,2}$+$I_{2,2}$+$I_{3,2}$)–
($M_{1,3}$+$M_{2,3}$+$M_{3,3}$+$T_{2,3}$+$T_{3,3}$+$H_{1,3}$+$H_{2,3}$+$I_{1,3}$+$I_{2,3}$) = -3 = 0

$U_{3,3}$ = 3, $H_{3,3}$ = 3

$W_{3,2}$ = ($M_{1,3}$+$M_{2,3}$+$M_{3,3}$+$T_{2,3}$+$T_{3,3}$+$H_{1,3}$+$H_{2,3}$+$H_{3,3}$+$I_{1,3}$+$I_{2,3}$) –
($M_{2,2}$+$M_{3,2}$+$M_{4,2}$+$T_{2,2}$+$T_{3,2}$+$T_{4,2}$+$H_{1,2}$+$H_{2,2}$+$H_{3,2}$+$I_{1,2}$+$I_{2,2}$+$I_{3,2}$) = 5

Finally, the mathematical equation (4.4d) developed for makespan determination with transfer and setup time for UIS/UW is used to calculate the makespan of the production sequence ABCD with transfer and setup time for FIS/UW as shown below.

$$Makespan = \sum_{j=1}^{m} M_{1,j} + \sum_{i=2}^{n} M_{i,m} + \sum_{i=1}^{n-1} H_{i,m} + \sum_{i=2}^{n} T_{i,m} = 61 \text{ hours} \qquad (4.4d)$$

The makespan obtained with the transfer and setup time between the stages of all the products for the production sequence ABCD in the given batch process with FIS/UW is illustrated using Gantt chart in Figure 5.7.

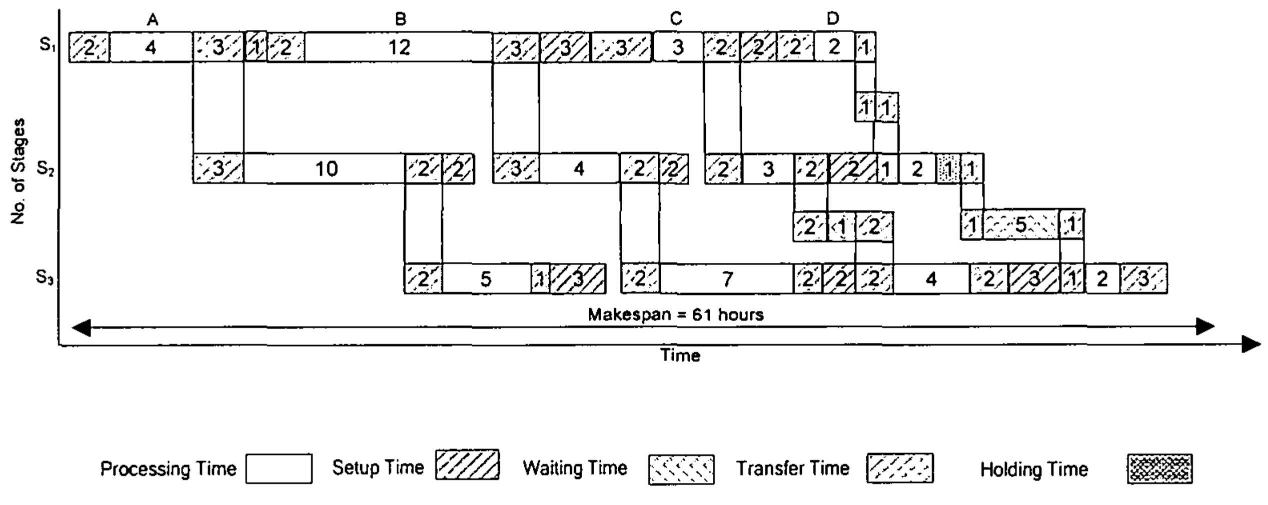


Figure 5.7: Gantt chart for production sequence ABCD with transfer and setup time for example 4.10

Figure 5.8 below illustrates the configuration of the batch process under study with the transfer and setup time for the production sequence ABCD.

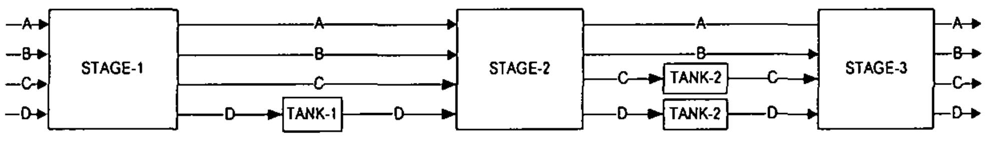


Figure 5.8: Temporary storage arrangement for production sequence ABCB for example 4.10

The result of equation (4.4e) are similar to the results obtained earlier in the case of ZW, NIS/UW and UIS/UW transfer policies i.e., the makespan of a batch process would increase with the inclusion of transfer and setup time.

5.5 MIS/UW transfer policy

The production sequence in example 4.12 i.e., ABCD with process makespan of 33 hours, is used to illustrate the recalculation of makespan with transfer and setup time for the given batch process under MIS/UW. For ease of calculation, the transfer and setup time data for the respective products is arranged in rows and columns as shown in Tables 5.12 and 5.13 below. The variables, T and U, are used to represent the transfer and setup time respectively in the forthcoming calculations.

Table 5.12
Stage transfer time for production sequence ABCD for example 4.12

Products	Transfer time (hour) T					
		columns (j)				
	rows (i)	0	1	2	3	4
A	1	2	3	2	2	2
B	2	2	1	1	2	3
C	3	3	2	2	1	2
D	4	2	3	2	2	1

Table 5.13
Stage setup time for production sequence ABCD for example 4.12

Products	Setup time (hour) U				
		columns (j)			
	rows (i)	1	2	3	4
A & B	1	2	3	1	2
B & C	2	1	2	3	2
C & D	3	5	5	3	2

Transfer and Setup Time

For the determination of makespan for MIS/UW in Chapter 4, it was assumed that batch process observes NIS/UW for stages 1 and 2 whereas UIS/UW for stage 3. Hence, the equations developed to incorporate transfer and setup time for NIS/UW and UIS/UW earlier could be modified for MIS/UW as shown below:

For NIS/UW from stage '1' to stage 'm-2':

$$V_{i,1}=0,\ \ I_{i,m}=0 \qquad i=1.......n\text{-}1$$

$$V_{1,j+1}=(V_{1,j}+M_{2,j}+T_{2,j-1})-(M_{1,j+1}+T_{1,j+1}) \qquad j=1........m-2 \qquad (4.5c)$$

$$I_{1,j}=(H_{1,j+1}+M_{1,j+1}+T_{1,j+1})-(H_{1,j}+M_{2,j}+T_{2,j-1}) \qquad j=1...........m\text{-}2 \qquad (5.1a)$$

$$V_{i,j+1}=(V_{i,j}+M_{i+1,j}+T_{i+1,j-1})-(M_{i,j+1}+T_{i,j+1}+I_{i-1,j+1})\ j=1........m-2,\ \ i=2.......n\text{-}1 \qquad (4.6c)$$

$$I_{i,j}=(H_{i,j+1}+M_{i,j+1}+T_{i,j+1}+I_{i-1,j+1})-(H_{i,j}+M_{i+1,j}+T_{i+1,j-1})$$
$$j=1.......m\text{-}2,\quad i=2.......n\text{-}1 \qquad (5.2a)$$

For UIS/UW at stage 'm-1':

$$V_{1,j}=(M_{2,j-1}+V_{1,j-1})-M_{1,j} \qquad j=m \qquad (4.7a)$$

$$\begin{bmatrix} \mathit{if}\ H_{1,j+1}>V_{1,j+1},\ \ W_{1,j}=(H_{1,j+1}+M_{1,j+1})-(H_{1,j}+M_{2,j}+T_{2,j})\ \ \mathit{else}\ \ W_{1,j}=0 \\ \\ \mathit{if}\ (H_{1,j+1}>V_{1,j+1}\ \ \mathit{and}\ \ W_{1,j}\le 0),\quad H_{1,j+1}=H_{1,j+1}+(-W_{1,j})\ \ \mathit{and}\ \ W_{1,j}=0\ (\mathit{pass}) \end{bmatrix}$$
$$j=m-1 \qquad (5.3a)$$

$$V_{i,j}=(\sum_{k=2}^{i+1}M_{k,j-1}+\sum_{k=2}^{i}T_{k,j-1}+\sum_{k=1}^{i}H_{k,j-1})-(\sum_{k=1}^{i}M_{k,j}+\sum_{k=2}^{i}T_{k,j}+\sum_{k=1}^{i-1}H_{k,j})$$
$$j=m,\quad i=2........n-1 \qquad (4.8c)$$

$$\begin{bmatrix} \mathit{if}\ H_{i,j}>V_{i,j},\ W_{i,j-1}=(\sum_{k=1}^{i}M_{k,j}+\sum_{k=2}^{i}T_{k,j}+\sum_{k=1}^{i}H_{k,j})-(\sum_{k=2}^{i+1}M_{k,j-1}+\sum_{k=2}^{i+1}T_{k,j-1}+\sum_{k=1}^{i}H_{k,j-1}) \\ \\ \mathit{else}\ W_{i,j-1}=0 \\ \\ \mathit{if}\ (H_{i,j}>V_{i,j}\ \ \mathit{and}\ \ W_{i,j-1}\le 0),\quad H_{i,j}=H_{i,j}+(-W_{i,j-1})\ \ \mathit{and}\ \ W_{i,j-1}=0\ (\mathit{pass}) \end{bmatrix}$$
$$j=m,\quad i=2........n-1 \qquad (5.4a)$$

For illustration purpose, the production sequence i.e., ABCD from example 4.12 is arranged to form a matrix with the introduction of variables, V, I and W as shown below.

$$\begin{array}{ccccc}
 & \multicolumn{2}{c}{\text{NIS/UW}} & \multicolumn{2}{c}{\text{UIS/UW}} \\
 & 1 & 2 & 3 & 4 \\
1 & 5 & 8 & 6 & 4 \\
 & V_{1,1} & V_{1,2} & V_{1,3} & V_{1,4} \\
2 & 9\ I_{1,1} & 3\ I_{1,2} & 2\ W_{1,3} & 2 \\
 & V_{2,1} & V_{2,2} & V_{2,3} & V_{2,4} \\
3 & 4\ I_{2,1} & 5\ I_{2,2} & 3\ W_{2,3} & 4 \\
 & V_{3,1} & V_{3,2} & V_{3,3} & V_{3,4} \\
4 & 4\ I_{3,1} & 4\ I_{3,2} & 2\ W_{3,3} & 2
\end{array}$$

The variables, V and I, for stages 1 and 2 are determined using equations (4.5c), (5.1a), (4.6c) and (5.2a) as demonstrated below using the same procedure as earlier developed for NIS/UW.

$V_{1,1} = 0 \quad U_{1,1} = 2 \quad H_{1,1} = 2$

$V_{2,1} = 0 \quad U_{2,1} = 1 \quad H_{2,1} = 1$

$V_{3,1} = 0 \quad U_{3,1} = 5 \quad H_{3,1} = 5$

$V_{1,2} = (H_{1,1}+M_{2,1}+T_{2,0}) - (M_{1,2}+T_{1,2}) = 3,$ $\qquad U_{1,2}=3, \quad H_{1,2}= 3$

$I_{1,1} = (H_{1,2} + M_{1,2}+T_{1,2}) - (H_{1,1}+M_{2,1}+T_{2,0}) = 0$

$V_{1,3} = (H_{1,2} + M_{2,2}+T_{2,1}) - (M_{1,3}+T_{1,3}) = -1 = 0$ $\qquad U_{1,3}=1, \quad H_{1,3}= 1$

$I_{1,2} = (H_{1,3} + M_{1,3}+T_{1,3}) - (H_{1,2} + M_{2,2}+T_{2,1}) = 2$

$V_{2,2} = (H_{2,1} + M_{3,1}+T_{3,0}) - (M_{2,2} +T_{2,2}+ I_{1,2}) = 2$ $\quad U_{2,2}=2,\ H_{2,2}=2$

$I_{2,1} = (H_{2,2} + M_{2,2} +T_{2,2}+ I_{1,2}) - (H_{2,1} + M_{3,1}+T_{3,0}) = 0$

$V_{2,3} = (H_{2,2} + M_{3,2}+T_{3,1}) - (M_{2,3} +T_{2,3}+I_{1,3})= 5$ $\quad U_{2,3}=3,\ H_{2,3}=5$

$I_{2,2} = (H_{2,3} + M_{2,3} +T_{2,3}+ I_{1,3}) - (H_{2,2} + M_{3,2}+T_{3,1}) = 0$

$V_{3,2} = (H_{3,1} + M_{4,1}+T_{4,0}) - (M_{3,2} +T_{3,2}+ I_{2,2}) = 4$ $\quad U_{3,2}=5,\ H_{3,2}=5$

$I_{3,1} = (H_{3,2} + M_{3,2} +T_{3,2}+ I_{2,2}) - (H_{3,1} + M_{4,1}+T_{4,0}) = 1$

$V_{3,3} = (H_{3,2} + M_{4,2}+T_{4,1}) - (M_{3,3} +T_{3,3}+I_{2,3}) = 8$ $\quad U_{3,3}=3,\ H_{3,3}= 8$

$I_{3,2} = (H_{3,3} + M_{3,3} +T_{3,3}+ I_{2,3}) - (H_{3,2} + M_{4,2}+T_{4,1}) = 0$

The variables, V and W, for stages 3 are determined using equations (4.7a), (5.3a), (4.8c) and (5.4a) as demonstrated below using the same procedure as earlier developed for UIS/UW. For this purpose, the process recipe would be changed to include the transfer time as follows:

$$\begin{array}{ccccc}
 & 1 & 2 & 3 & 4 \\
1 & T_{1,0}+5+T_{1,1} & 8+T_{1,2} & T_{1,3}+6 & T_{1,4}+4 \\
2 & T_{2,0}+9 & 3+T_{2,1} & T_{2,2}+2 & T_{2,3}+2 \\
3 & T_{3,0}+4 & 5+T_{3,1} & T_{3,2}+3 & T_{3,3}+4 \\
4 & T_{4,0}+4 & 4+T_{4,1} & T_{4,2}+2 & T_{4,3}+2
\end{array}$$

Next, the variables, V, used to represent idle times between process stages and waiting time, W, of product intermediate inside the temporary storage, are introduced in the resulting matrix after the addition of transfer times to the processing times above.

	1	2	3	4
1	10	10	8	6
			$V_{1,3}$	$V_{1,4}$
2	11	4	3 $W_{1,3}$	4
			$V_{2,3}$	$V_{2,4}$
3	7	7	5 $W_{2,3}$	5
			$V_{3,3}$	$V_{3,4}$
4	6	7	4 $W_{3,3}$	4

The calculation of variables, V and W, between stages 3 and 4 is as follows:

$V_{1,4} = (M_{2,3}+H_{1,3}) - M_{1,4} = -2 = 0$, $U_{1,4}=2$, $H_{1,4}= 2$

$W_{1,3} = (H_{1,2} + M_{1,2}) - (H_{1,1}+ M_{2,1}+T_{2,1}) = 2$

$V_{2,4} = (M_{2,3}+M_{3,3}+T_{2,3}+H_{1,3}+H_{2,3}) - (M_{1,4}+M_{2,4}+T_{2,4}+H_{1,4})= 1$ $U_{2,4}=2,$ $H_{2,4}=2$

$W_{2,3} = (M_{1,4}+M_{2,4}+T_{2,4}+H_{1,4}+H_{2,4}) - (M_{2,3}+M_{3,3}+T_{2,3}+T_{3,3}+H_{1,3}+ H_{2,3}) = 0$ (pass)

$V_{3,4}= (M_{2,3}+M_{3,3}+M_{4,3}+T_{2,3}+T_{3,3}+H_{1,3}+H_{2,3}+H_{3,3}) - (M_{1,4}+M_{2,4}+M_{3,4}+T_{2,4}+T_{3,4}+H_{1,4}+H_{2,4})$
$= 5$

$U_{3,4}=2$, $H_{3,4}=5$, $W_{3,3} = 0$

Finally, the mathematical equation (4.4d) developed for makespan determination for UIS/UW with transfer and setup time is used to calculate the makespan of production sequence ABCD with transfer and setup time for MIS/UW transfer policy as shown below.

$$Makespan = \sum_{j=1}^{m} M_{1,j} + \sum_{i=2}^{n} M_{i,m} + \sum_{i=1}^{n-1} H_{i,m} + \sum_{i=2}^{n} T_{i,m} = 62 \text{ hours} \qquad (4.4d)$$

The illustration of transfer and setup time location between the stages of all the products for the production sequence ABCD of the given batch process recipe with MIS/UW transfer policy is done using Gantt chart in Figure 5.9.

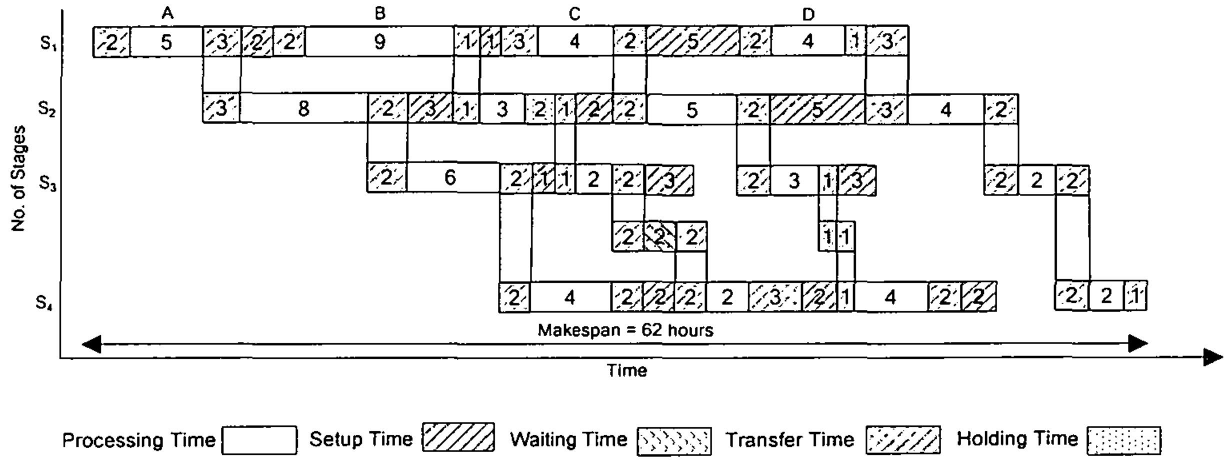


Figure 5.9: Gantt chart for production sequence ABCD with transfer and setup time for example 4.12

Number of Storage Tanks

Next, the number of temporary storage required is determined. Since the values of $W_{1,3}$ is greater than zero and $W_{2,3}$ is zero (pass), they indicate the need of two temporary storages after the third stage. The zero value obtained in case of $W_{2,3}$ shows that product has only to pass through the temporary storage before transferring to the next stage. The batch process arrangement in this scenario is illustrated in Figure 5.10.

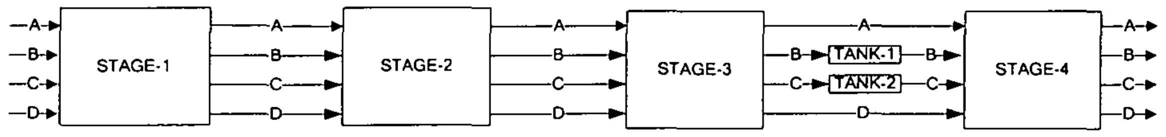


Figure 5.10: Temporary storage arrangement for production sequence ABCD for example 4.12

The makespan obtained above is more than obtained earlier for MIS/UW transfer policy without transfer and setup time in chapter 4. It clearly indicates that it increases when determined with transfer and setup time.

5.5 Summary

For industrial applications, the time to transfer the product intermediate from one stage to the next and the setup time for any stage prior to accepting the product intermediate from earlier stage must be considered for scheduling of batch processes. It is obvious that the makespan of the given batch process for a specific production sequence would increase with the inclusion of transfer and setup time. Similar to the work done in the relevant literature for the scheduling of batch processes, the present work has also considered the sequence dependent transfer and setup time. This means that the available data for transfer and setup time could vary with the variation in production sequence.

The transfer and setup time can be incorporated in the proposed procedure for the determination of makespan using matrix representation. For the illustration purpose, the developed procedure has been applied on example batch process recipes for each transfer policy. The conclusion drawn from the makespan results with transfer and setup time confirms the fact that prior to the determination of optimal production sequence, calculation of makespan is important as it allows screening to be done. A summary of the mathematical formulations developed for makespan determination with transfer and setup times for various transfer policies is shown in Appendix C.

CHAPTER 6

OPTIMAL SOLUTION SCREENING

The purpose of conducting the screening for optimal solution is to determine the desired batch process sequence according to the sets of criteria specified by the process planner. The criteria may either be minimum makespan, inter-stage idle time or number of temporary storages required. The optimal solution for the same batch process recipe could be different for different transfer policies. The choice of a transfer policy depends on the type of products to be manufactured and the time period during which the production targets have to be achieved.

6.1 Optimal solution screening procedure

The developed method for determining makespan using matrix representation could be applied on all possible production sequences generated by rearranging the batch process sequences. This enables the developed method to produce a range of possible solutions from which a process planner could choose according to various criteria described above. The developed method is also capable to produce data other than makespan determination for each production sequence. This includes determination of inter-stage idle time, holding time of product intermediates inside the same stages and waiting time of product intermediates inside the temporary storages as per transfer policy adopted.

In order to speed up the calculation, the proposed method is programmed on an Intel Pentium® IV CPU 2.40 GHz machine using Microsoft Visual C++ ™. A flowchart for the screening of optimal solution on the basis of the production sequence with minimum makespan is shown in Figure 6.1. The programming code is shown in Appendix E. A sample screen output of developed programming code is also shown in Appendix F.

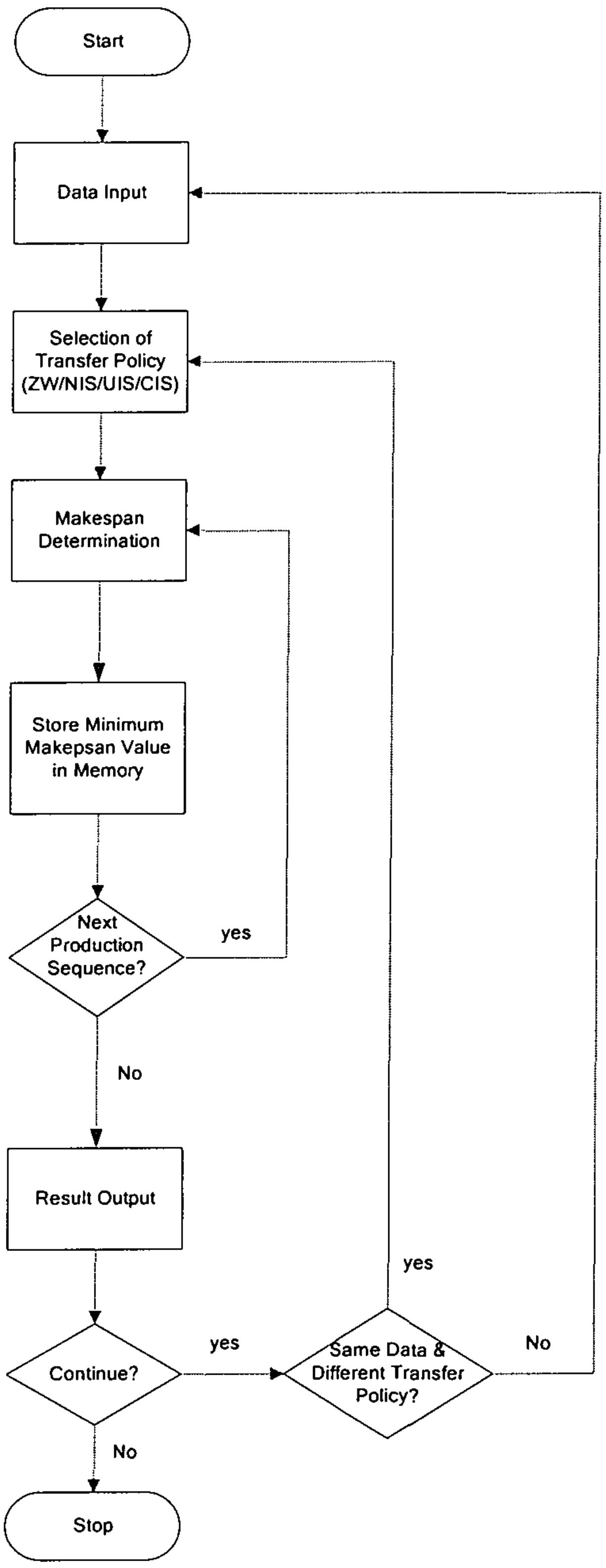

Start
Data Input
Selection of Transfer Policy (ZW/NIS/UIS/CIS)
Makespan Determination
Store Minimum Makepsan Value in Memory
Next Production Sequence?
yes
No
Result Output
Continue?
yes
No
Stop
Same Data & Different Transfer Policy?
yes
No

Figure 6.1: Flowchart for optimal solution screening

The application of optimal solution screening for batch processes is shown below using a batch process operated under ZW transfer policy.

ZW transfer policy

For the purpose of demonstrating the optimal solution screening procedure for a batch process operated under ZW transfer policy, the example 4.3 is selected.

$$\begin{array}{cccc}
 & S_1 & S_2 & S_3 \\
A & 10 & 20 & 5 \\
 & V_{1,1} & V_{1,2} & V_{1,3} \\
B & 15 & 8 & 12 \\
 & V_{2,1} & V_{2,2} & V_{2,3} \\
C & 20 & 7 & 9
\end{array}$$

The data for sequence dependent transfer and setup time is given in Table 6.1 and 6.2 below for every possible sequence of the products in example 4.3.

Table 6.1. Stage transfer time for products in example 4.3

Products	T_0	T_1	T_2	T_3
A	3	2	2	1
B	2	3	2	2
C	2	3	2	2

Table 6.2. Stage setup time for products in example 4.3

Product Sequence	U_1	U_2	U_3
A-B	1	3	2
A-C	5	5	3
B-A	6	4	2
B-C	4	1	2
C-A	4	1	2
C-B	2	3	3

The advantage of the developed method can be observed from the results shown in Table 6.3. Table 6.3 shows the calculated makespan and the corresponding variable "H" representing the inter-stage idle time + stage setup time for all the possible production sequences. Also, it is noticed that the inter-stage idle time + stage setup time varies for different production sequences. This is because of the change of position of products in every production sequence.

Table 6.3
Makespan and inter-stage idle time with sequence dependent transfer and setup time for all possible production sequences for example 4.3

Production Sequence	Makespan (hour)	Inter-Stage Idle Time + Stage Setup Time (hour)					
		$H_{1,1}$	$H_{1,2}$	$H_{1,3}$	$H_{2,1}$	$H_{2,2}$	$H_{2,3}$
ABC	92	8	3	8	4	16	12
ACB	91	5	5	9	2	10	10
BAC	91	6	9	17	5	5	9
BCA	96	4	16	12	4	8	19
CAB	96	4	8	19	8	3	8
CBA	96	2	10	10	6	9	17

It can be observed from Table 6.3 that there are two production sequences that could be chosen as optimal based on the criteria of minimum makespan i.e., ACB and BAC with makespan of 91 hours.

However, if additional criteria be used for the selection, the production sequences can be ranked accordingly for process planner to select from the generated solutions. The additional criterion could either be the inter-stage idle time or the production sequence. Supposed for the present example, a product can not be processed immediately before or after a specific product due to some process requirements. For example, in Table 6.3, assuming that product A could not be processed immediately after product B. The optimal sequence BAC will no longer be the feasible solution. Hence, among the optimal solutions, the only solution left that meets the criteria is second optimal solution i.e. ACB.

The same procedure for optimal solution screening could be applied in case of the batch processes operated under UW transfer policy with various intermediate storage configurations discussed earlier. In addition to the determination of makespan and inter-stage idle times as in ZW transfer policy, the optimal solution screening procedure could also determine the number and location of temporary storages in UIS/UW, FIS/UW, MIS/UW and the holding time inside the same stage in the case of NIS/UW transfer policies for every possible sequence of the given batch process recipe.

6.2 Comparison of proposed method with available mathematical methods

In scheduling practice, the efficiency of any scheduling technique is usually measured in terms of CPU time consumed to determine the optimal solution. It has also been observed that CPU time increases exponentially with problem size (Pitty and Karimi, 2008). To demonstrate this, simulation studies using a computer code developed in Microsoft Visual C++^{TM} for matrix representation have been performed for each transfer policy stated earlier. The results so obtained are shown in Appendix D. The simulation studies have been performed using a CPU configuration similar to Pitty and Karimi, 2008 i.e., Intel Pentium$^{®}$ IV CPU 2.4 GHz. The problem sizes and batch process recipe data have been generated randomly using the same distribution range adopted by Pitty and Karimi (2008) for processing time (24, 240) and setup time (0, 24). However, the data distribution range for transfer time is (1, 24). For comparison among various problem sizes, the arithmetic mean of CPU time is also calculated.

The CPU time observed for various problem sizes is fairly similar to the observation of Pitty and Karimi, 2008. It has been observed that for problem size (n=10, m=5), the CPU time obtained is less than 2100 sec which is found better than two of the mathematical models that were unable to produce the optimal solution within 5000 sec as reported in Pitty and Karimi (2008). However for problem size (n=12, m=5) and larger, the proposed method was not able to produce the optimal solution within 5000 sec. The similar observation was made by Pitty and Karimi (2008) where none of the 18 mathematical models was able to produce optimal solution within 5000 sec for problem size of (n=12, m=5).

It is important to note that the CPU time used to solve each of the problem that varies in size as shown in Appendix B is based on using total enumeration approach. For each problem, the makespan for all the possible sequences are determined separately while screening for the optimal solution. The simulation results also reiterate the fact that with increase in problem size, CPU time also increases. A possible way to reduce the CPU time in finding the optimal solution is to apply partial enumeration instead of total enumeration. In partial enumeration, the number of sequences to be evaluated is less compared to the total enumeration approach, therefore reducing the CPU time. However, the selection of the sequences to evaluated should be such that the optimal or near optimal sequence(s) be included in the selected sequences. For this purpose, a set of heuristic rules are proposed. These rules are developed based on some critical observations made on the matrix arrangement. It will be shown later that the application of these rules has resulted in providing the optimal solution based on the minimum makespan criteria for all the example problems tested.

6.3 Development of the heuristic rules

The development of the heuristic rules has been done based on two critical observations made on the matrix representation of the batch process recipe over all the examples presented earlier.

1. The optimal sequence can start with the product that has the least makespan in the first stage.

2. The optimal sequence can start with the product that has the sum of its processing recipe and processing time in the last stages of all other products with the least value compared to the value when calculated for other products using the same procedure

Based on the observations, all the potential sequences that will result in the optimal solution are identified. The application of the heuristic rules developed is demonstrated in the example below after the inclusion of transfer time.

Example (example 4.3 - ZW transfer policy)

In this example, the batch process recipe from example 4.3 is analyzed after the inclusion of respective transfer time of each product from Table 5.1.

	S_1	S_2	S_3
A	13	22	8
B	17	11	16
C	22	10	13

1. Product A has the least processing time for its first stage i.e. 13, compared to other products. Therefore, it can be said that the optimal sequence with minimum makespan could be the one that has product A as the first product in the sequence.

2. Determine the sum of processing recipe of the first product and processing time in the last stages of all other products. Repeat the same procedure for second and third product as shown below.

   Consider product A is placed first in the common path.
   $AS_1+AS_2+\underline{AS_3+BS_3+CS_3} = 13+22+8+16+13 = 72$

   Consider product B is placed first in the common path.
   $BS_1+BS_2+ \underline{AS_3+BS_3+CS_3} = 17+11+8+16+13 = 65$

   Consider product C is placed first in the common path.
   $CS_1+CS_2+ \underline{AS_3+BS_3+CS_3} = 22+10+8+16+13 = 69$

It is observed from point 2 that the sequence giving the minimum value is the one that is starting from product B. Hence, it can be said that the optimal solution with minimum makespan could be the one that has product B as the first product in the sequence.

From the observations made in point 1 and 2 above, it can be concluded that optimal production sequence would be the one that has either product A or B as the first product in the sequence. Therefore, the enumeration is carried out only for product A and B to produce the production sequences that would also include the optimal sequence with least makespan. Hence the total number of production sequences generated would be four i.e. two with product A and two with product B as first product in the sequences. This can also be conferred from the results of total enumeration in Table 6.3 which shows that there are two optimal sequences ACB and BAC with least makespan value.

Heuristic Rules:

Step 1: $Min\ (P_{i,1}) \qquad (i = 1 \ldots\ldots\ldots. n)$

Find the product that has the least processing time in the first stage

Step 2: $P_i = \sum_{j=1}^{m-1} S_{i,j} + \sum_{k=1}^{n} S_{k,m}$

Detemine the sum of processing recipe of the first product and the processing time in the last stages of all other products. Repeat the same procedure with second and third product

Step 3: $Min\ (P_i)$

Find the minimum value from the values calculated in step 2 and the corresponding product

Condition 1. $\quad if \ 'i' \ in \ P_i \ and \ P_{i,1} \ is \ same$

If the same product is found in step 1 and 3, the partial enumeration would be carried out to produce only those sequences that place this product as the first product in the sequence

$$Number\ of\ enumeration\ \text{needed}: \ \frac{n!}{n}\times 1$$

Condition 2. $\quad if \ 'i' \ in \ P_i \ and \ P_{i,1} \ is \ different$

If the product(s) found in step 3 differs from step 1, the partial enumeration should have all the sequences begining with all the products identified in step 1 and 3 as the potential sequences to be screened.

$$Number\ of\ enumeration\ \text{needed}: \ \frac{n!}{n}\times (no.\ of\ times, the'i'is\ different)$$

The following additional examples are presented to demonstrate further the application of the developed heuristic rules. The examples have been solved for ZW transfer policy with computer code developed for this purpose in Microsoft Visual C++™ for matrix approach on an Intel Pentium® IV CPU 2.40 GHz.

1. n = 4, m = 4

	S_1	S_2	S_3	S_4	SUM
P_1	14	45	49	37	239
P_2	36	11	37	44	215
P_3	29	35	50	30	245
P_4	45	30	19	20	225

Minimum S_1 = 14, Minimum SUM = 215

Possible optimal production sequences = Starting with products P_1 or P_2

Total enumeration possible = 24

Enumeration recommended by heuristics= 12 i.e. 6 with each of P_1 and P_2

Reduction in solution search space = 50%

Optimal production sequence obtained = $P_2P_1P_3P_4$ = 244 hours

2. n = 7, m = 4

	S_1	S_2	S_3	S_4	SUM
P_1	46	16	21	44	340
P_2	22	18	27	45	324
P_3	33	45	26	26	361
P_4	30	40	24	43	351
P_5	44	30	18	15	349
P_6	10	31	42	35	340
P_7	39	40	19	49	355

Minimum S_1 = 10, Minimum SUM = 324

Possible optimal production sequences = Starting with products P_2 or P_6

Total enumeration possible = 5040

Enumeration recommended by heuristics = 1440 i.e. 720 with each of P_2 and P_6

Reduction in solution search space = 71%

Optimal production sequence obtained = $P_2P_1P_6P_4P_7P_3P_5$ = 335 hours

3. n = 8, m = 6

	S_1	S_2	S_3	S_4	S_5	S_6	SUM
P_1	21	24	44	26	19	14	325
P_2	18	11	32	31	11	17	294
P_3	38	18	20	25	26	25	318
P_4	34	12	24	47	41	12	349
P_5	11	22	38	30	26	14	318
P_6	17	26	47	27	45	49	353
P_7	45	49	13	29	34	18	361
P_8	25	36	11	28	14	42	305

Minimum $S_1 = 11$, Minimum SUM = 294

Possible optimal production sequences = Starting with products P_5 or P_2

Total enumeration possible = 40320

Enumeration recommended by heuristics = 10080 i.e. 5040 with each of P_5 and P_2

Reduction in solution search space = 75%

Optimal production sequence obtained = $P_5P_6P_4P_1P_7P_8P_3P_2$ = 417 hours

4. n = 9, m = 6

	S_1	S_2	S_3	S_4	S_5	S_6	SUM
P_1	26	23	39	27	28	34	362
P_2	32	30	16	12	17	28	326
P_3	20	32	34	17	25	16	347
P_4	15	11	11	32	32	15	320
P_5	49	21	45	32	49	12	415
P_6	19	17	41	23	13	20	332
P_7	26	45	38	28	20	40	376
P_8	42	38	41	29	33	12	402
P_9	43	12	21	25	35	42	355

Minimum $S_1 = 15$ Minimum SUM = 320

Possible optimal production sequences = Starting with products P_4

Total enumeration possible = 362880

Enumeration recommended by heuristics = 40320 with each P_4

Reduction in solution search space = 88%

Optimal production sequences obtained:

$P_4P_3P_9P_1P_5P_7P_8P_6P_2$, $P_4P_3P_9P_1P_7P_5P_8P_6P_2$,

$P_4P_6P_9P_1P_5P_7P_8P_3P_2$, $P_4P_6P_9P_1P_7P_5P_8P_3P_2$ = 449 hours

5. n = 10, m = 7

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	SUM
P_1	22	45	11	17	46	27	35	434
P_2	38	29	32	28	17	37	25	447
P_3	20	49	13	50	35	33	20	466
P_4	22	45	43	44	50	43	25	513
P_5	35	23	45	29	10	33	24	441
P_6	30	14	16	21	44	49	19	440
P_7	20	15	15	47	39	15	14	417
P_8	32	49	33	21	34	12	38	447
P_9	40	40	46	45	39	36	46	512
P_{10}	13	37	29	36	46	13	20	440

Minimum $S_1 = 13$ Minimum SUM = 417

Possible optimal production sequences = Starting with products P_7 or P_{10}

Total enumeration possible = 3628800

Enumeration recommended by heuristics = 725760 i.e. 362880 with each of P_7 and P_{10}

Reduction in solution search space = 80%

Optimal production sequence obtained = $P_6P_{10}P_5P_4P_9P_3P_8P_2P_1P_7$ = 568 hours

Table 6.4. CPU time for total and partial enumeration for various problem sizes

Problem Size	No. of total enumeration	CPU time (sec)	No. of partial enumeration	CPU time (sec)
4x4	24	0.01	12	0.005
7x4	5040	1.642	1440	0.469
8x6	40320	21.017	10080	5.254
9x6	362880	211.989	40320	23.554
10x7	3628800	2090.657	725760	418.131

The optimal production sequence obtained in each of the above problem size has also been verified using total enumeration. Significant reduction in computational time could be observed from Table 6.4 for various problem sizes. Similar observation with regards to reduction in computational time has been observed for same problem sizes and batch process recipes in case of NIS/UW, UIS/UW, FIS/UW and MIS/UW transfer policies. This shows the benefit of using heuristic procedure in combination with partial enumeration that can limit the search space for optimal production sequence search.

It is observed that the developed heuristic rules have been able to produce the optimal solution for all the example problems studied above. However, the heuristic rules are not based on analytical approach. Therefore, there may be chances where the solution obtained using these heuristic rules for any specific batch process recipe could not be the one that meets the criteria of minimum makespan. However, it guarantees to obtain near optimal solution instead of optimal solution. For this purpose, an extension to the heuristic rules has been suggested to consider more products for enumeration thereby increasing the possibility of finding the optimal solution. The selection of more products could be done by finding the products that correspond to the values next to the minimum values determined earlier in step 1 and 3 of heuristic rules. The makespan calculated is then compared with the previous makespan value. The same procedure could be repeated till the new makespan value obtained is greater than the previous one. This has been illustrated using a schematic diagram below:

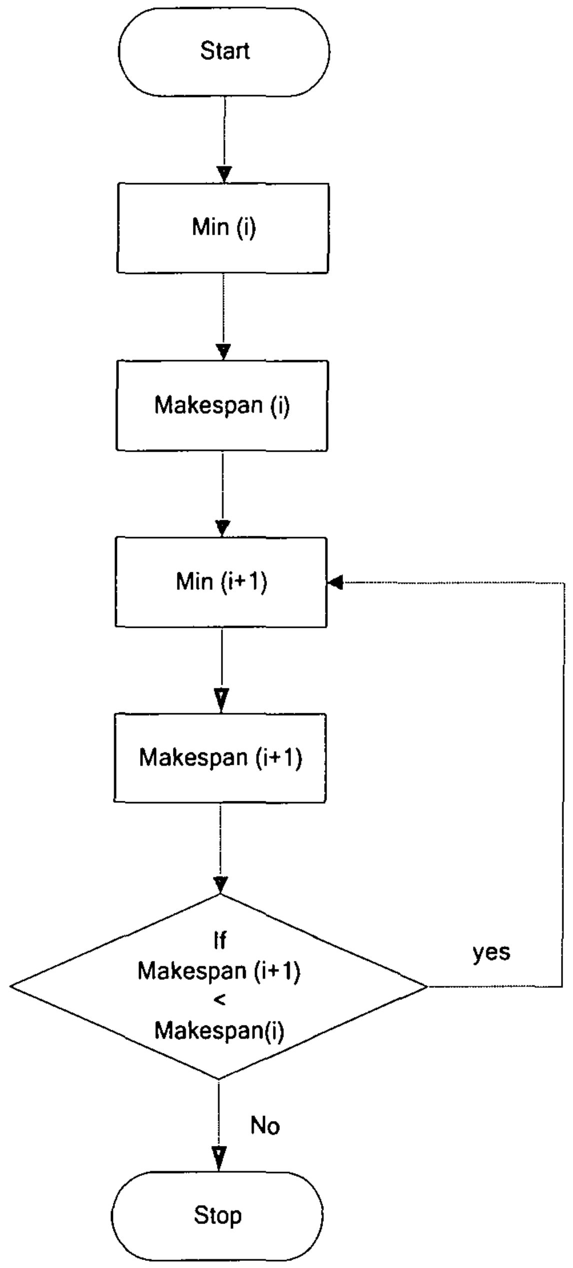


Figure 6.2: Flowchart for partial enumeration

It is obvious that the increase in number of products for enumeration would also increase the CPU time for searching the optimal solution. However, the need of further iteration would solely depend upon the process engineer's decision to find the optimal or near optimal solution. Some case studies from relevant literature have also been solved using matrix representation coupled with heuristics procedure proposed above to verify the accuracy of the proposed approach for determining the optimal production sequence.

6.4 Case Studies

The proposed procedure developed for scheduling of multiproduct batch processes appears to be relatively effective and gives promising results as demonstrated using the various examples. To further verify the fact, extended tests using more case studies particularly from the industry or published literature are conducted. For this purpose, the batch process data taken from Kim et al. (1996) and Jung et al. (1996) for different transfer policies are used. Moon et al. (1996), Pozivil and Zdansky (2001), and, Lee et al. (2002) have also referred to similar case studies in their work for developing new completion time algorithms for scheduling multiproduct batch processes.

In this work, the proposed procedure is applied on the case studies specified using the same system configuration used earlier for various problem sizes. The effectiveness of the proposed procedure is observed in two ways. The observation is firstly made on the results generated in addition to the makespan such as amount and location of inter-stage idle time for ZW, holding time of product intermediate within the same stage in NIS/UW and FIS/UW and number and location of temporary storages in UIS/UW. Secondly, the observation is on the CPU time for producing the optimal solution. The results obtained from case studies are in agreement with those obtained in the literature and Gantt charts drawn for each transfer policy.

6.4.1 Case Study 1 (Kim et al., 1996)

The batch process recipe for producing four products using four stages is given in Table 6.5. The corresponding stage transfer time and stage setup time are also given in the same table.

Table 6.5.Batch process recipe for four products and four stages for case study 1

	Stage Processing Time				Stage Transfer Time				
Product	S_1	S_2	S_3	S_4	T_0	T_1	T_2	T_3	T_4
P_1	10	20	5	30	2	2	2	2	3
P_2	15	8	12	10	3	3	3	3	1
P_3	20	7	9	5	2	4	2	2	1
P_4	13	7	17	10	2	2	1	4	2

	Stage Setup Time			
Product Sequence	U_1	U_2	U_3	U_4
P_1-P_2	3	1	2	4
P_1-P_3	2	2	1	3
P_1-P_4	1	4	2	2
P_2-P_1	4	1	2	3
P_2-P_3	1	1	4	3
P_2-P_4	3	2	3	2
P_3-P_1	2	1	4	3
P_3-P_2	1	2	3	2
P_3-P_4	2	2	2	2
P_4-P_1	4	3	4	3
P_4-P_2	1	4	3	3
P_4-P_3	3	2	2	1

Table 6.6 shows a summary of results for ZW, NIS/UW and UIS/UW transfer policies. The summary shows the optimal production sequence which meets the criteria of minimum makespan.

Table 6.6: Summary of results for optimal production sequence for case study 1

Transfer Policy	Optimal Production Sequence	Makespan (hour)	Slack Variables (hour)			CPU Time (sec)
			Inter-Stage Idle Time+ Stage Setup Time (H)	Holding Time (I)	Waiting Time (W)	
ZW	$P_1P_4P_2P_3$	130	$H_{1,1}=22$			0.015
			$H_{1,2}=15$			
			$H_{1,3}=17$			
			$H_{1,4}=2$			
			$H_{2,1}=3$			
			$H_{2,2}=13$			
			$H_{2,3}=3$			
			$H_{2,4}=6$			
			$H_{3,1}=1$			
			$H_{3,2}=12$			
			$H_{3,3}=8$			
			$H_{3,4}=8$			
NIS/UW	$P_1P_4P_2P_3$	126	$H_{1,1}=1$	$I_{1,1}=10$		0.031
			$H_{1,2}=4$	$I_{1,2}=0$		
			$H_{1,3}=6$	$I_{1,3}=11$		
			$H_{1,4}=2$	$I_{2,1}=0$		
			$H_{2,1}=1$	$I_{2,2}=13$		
			$H_{2,2}=11$	$I_{2,3}=0$		
			$H_{2,3}=3$	$I_{3,1}=2$		
			$H_{2,4}=6$	$I_{3,2}=7$		
			$H_{3,1}=1$	$I_{3,3}=0$		
			$H_{3,2}=1$			
			$H_{3,3}=4$			
			$H_{3,4}=4$			
UIS/UW	$P_1P_4P_3P_2$	120	$H_{1,1}=1$		$W_{1,1}=8$	0.032
			$H_{1,2}=4$		$W_{1,2}=0$	
			$H_{1,3}=6$		$W_{1,3}=7$	

$H_{1,4}=2$	$W_{2,1}=0$
$H_{2,1}=3$	$W_{2,2}=3$
$H_{2,2}=7$	$W_{2,3}=9$
$H_{2,3}=2$	$W_{3,1}=0$
$H_{2,4}=1$	$W_{3,2}=0$
$H_{3,1}=1$	$W_{3,3}=0$
$H_{3,2}=10$	
$H_{3,3}=5$	
$H_{3,4}=3$	

The results obtained in Table 6.6 using the proposed method are similar to the results obtained by Kim et al. (1996) using MILP and MINLP methods. Also, the proposed method appears to give reasonable CPU time requirements to produce optimum solution. In addition, the results also show other useful information such as idle time between process stages, waiting time inside the temporary storage and holding time within the same stage. The Gantt charts for optimal production sequence obtained above are shown in Figure 6.3, 6.4 and 6.5 respectively.

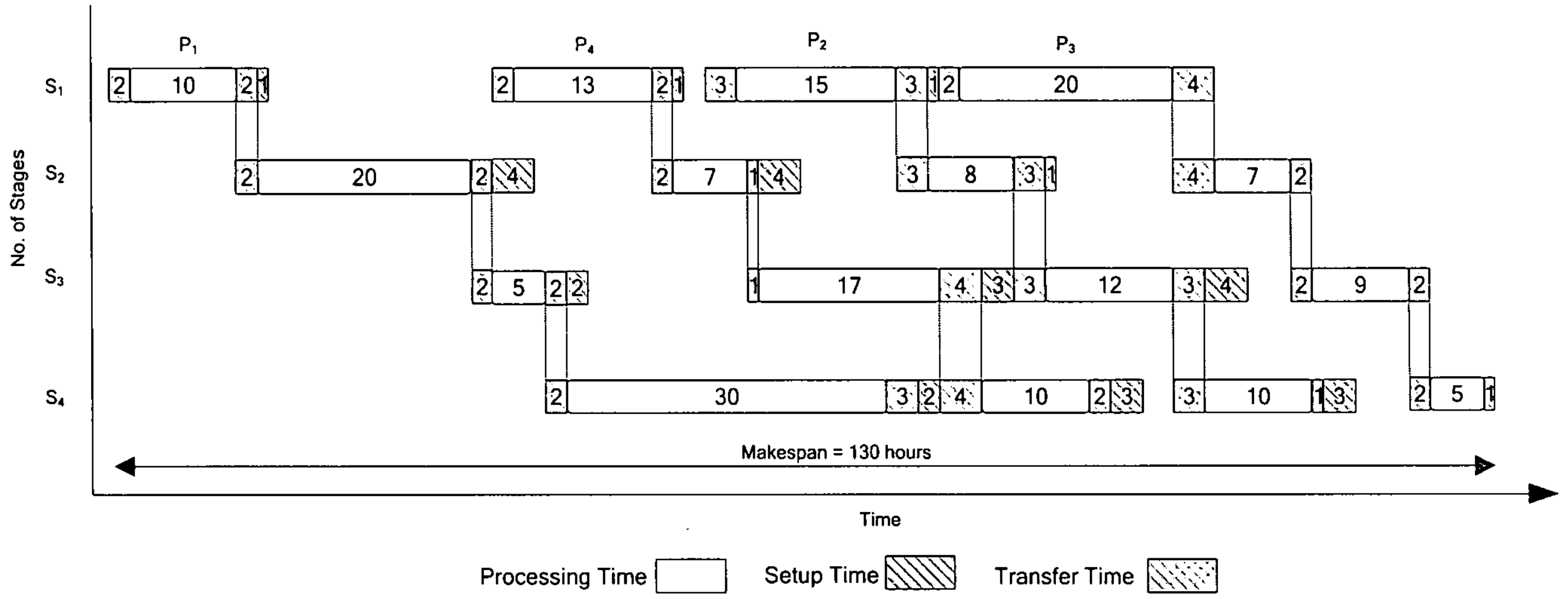


Figure 6.3 Gantt chart for production sequence $P_1P_4P_2P_3$ for case study 1 (ZW transfer policy)

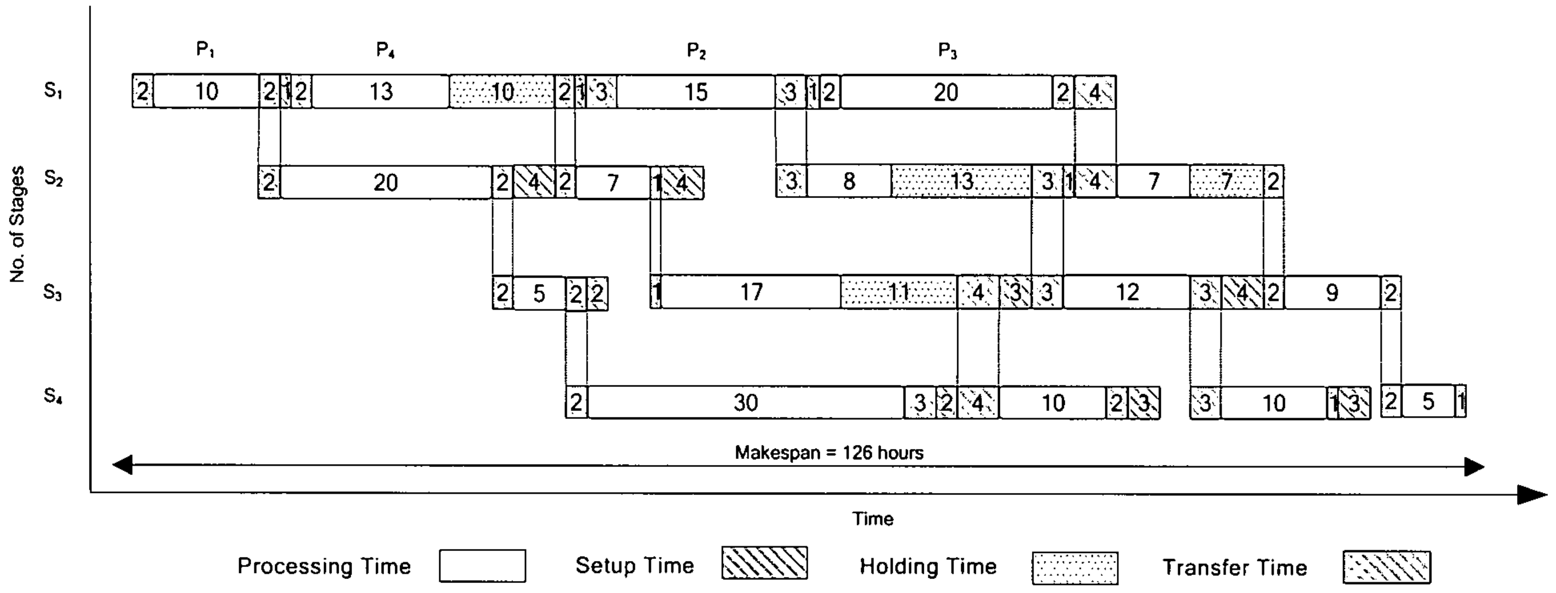


Figure 6.4 Gantt chart for production sequence $P_1P_4P_2P_3$ for case study 1 (NIS/UW transfer policy)

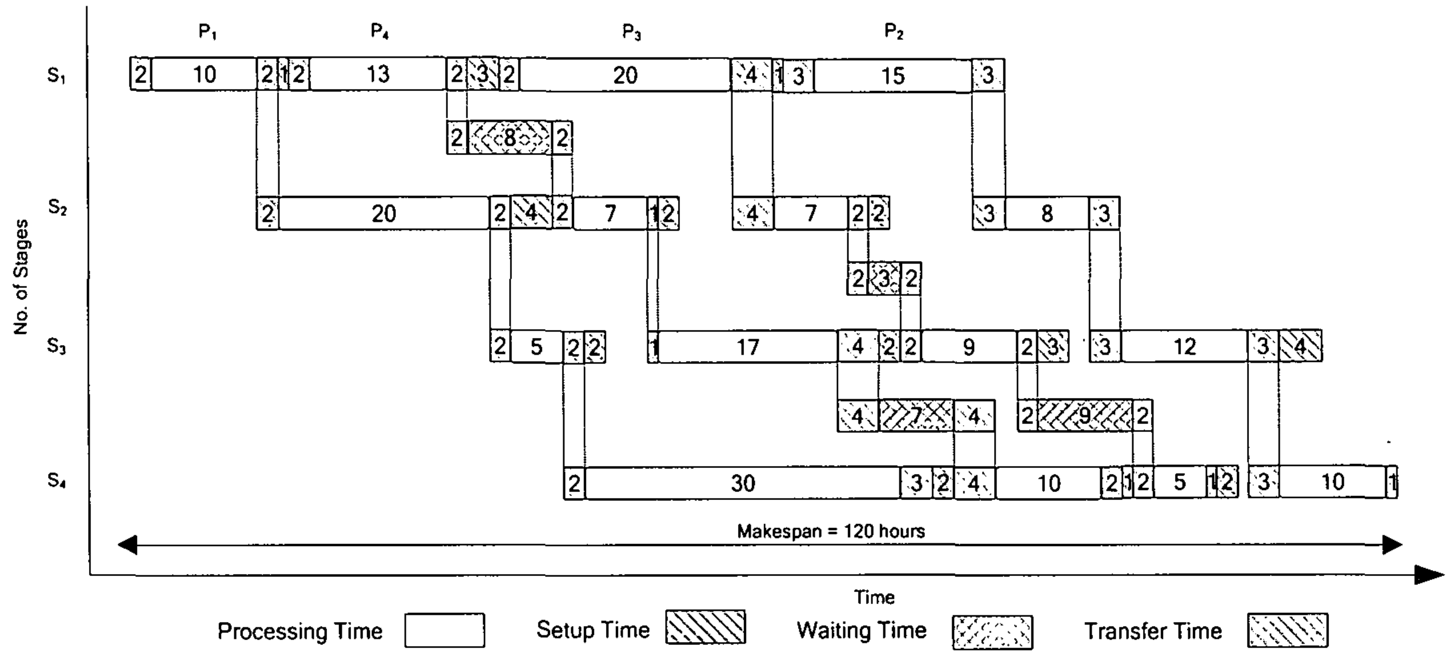


Figure 6.5 Gantt chart for production sequence $P_1P_4P_3P_2$ for case study 1 (UIS/UW transfer policy)

6.4.2 Case Study 2 (Jung et al., 1996)

The batch process recipe for producing four products using six stages is given in Table 6.7. The corresponding stage transfer time, stage setup time and storage setup time are also provided.

Table 6.7. Batch process recipe for four products and six stages for case study 2

	Stage Processing Time						Stage Transfer Time						
Product	S_1	S_2	S_3	S_4	S_5	S_6	T_0	T_1	T_2	T_3	T_4	T_5	T_6
P_1	10	15	20	12	8	11	2	2	2	2	3	2	1
P_2	15	8	12	10	9	13	3	3	3	3	1	1	2
P_3	10	22	9	5	6	9	2	4	2	2	1	2	2
P_4	20	12	7	10	10	4	2	2	1	4	2	2	1

	Stage Setup Time					
Product Sequence	U_1	U_2	U_3	U_4	U_5	U_6
P_1-P_2	3	1	2	4	2	3
P_1-P_3	2	2	1	3	1	4
P_1-P_4	1	4	2	2	3	2
P_2-P_1	4	1	2	3	2	2
P_2-P_3	1	1	4	3	1	1
P_2-P_4	3	2	3	2	2	1
P_3-P_1	2	1	4	3	3	1
P_3-P_2	1	2	3	2	2	2
P_3-P_4	4	2	2	2	2	1
P_4-P_1	4	3	4	3	1	2
P_4-P_2	1	4	3	3	4	2
P_4-P_3	3	2	2	1	2	3

	Storage Setup Time					
Product	G_1	G_2	G_3	G_4	G_5	G_6
P_1	1	2	2	2	1	4
P_2	2	2	1	4	1	2
P_3	2	3	3	1	2	1
P_4	1	3	2	3	2	2

The results of determining the optimal solution with minimum makespan for FIS/UW transfer policy are shown in Table 6.8.

Table 6.8: Summary of results for optimal production sequence for case study 2

Transfer Policy	Optimal Production Sequence	Makespan (hour)	Slack Variables (hour)			CPU Time (sec)
			Inter-Stage Idle Time + Stage Setup Time (H)	Holding Time (I)	Waiting Time (W)	
FIS/UW	$P_1P_2P_3P_4$	136	$H_{1,1}=3$	$I_{1,1}=0$	$W_{1,1}=0$	0.01
			$H_{1,2}=4$	$I_{1,2}=0$	$W_{1,2}=6$	
			$H_{1,3}=2$	$I_{1,3}=0$	$W_{1,3}$=pass	
			$H_{1,4}=5$	$I_{1,4}=0$	$W_{1,4}=0$	
			$H_{1,5}=8$	$I_{1,5}=0$	$W_{1,5}=0$	
			$H_{1,6}=6$	$I_{2,1}=0$	$W_{2,1}=0$	
			$H_{2,1}=1$	$I_{2,2}=0$	$W_{2,2}=0$	
			$H_{2,2}=2$	$I_{2,3}=0$	$W_{2,3}$=pass	
			$H_{2,3}=4$	$I_{2,4}=0$	$W_{2,4}$=pass	
			$H_{2,4}=3$	$I_{2,5}=0$	$W_{2,5}=6$	
			$H_{2,5}=1$	$I_{3,1}=0$	$W_{3,1}=0$	
			$H_{2,6}=1$	$I_{3,2}=0$	$W_{3,2}=0$	
			$H_{3,1}=4$	$I_{3,3}=0$	$W_{3,3}=0$	
			$H_{3,2}=2$	$I_{3,4}=0$	$W_{3,4}=0$	
			$H_{3,3}=5$	$I_{3,5}=0$	$W_{3,5}=0$	
			$H_{3,4}=5$			
			$H_{3,5}=10$			
			$H_{3,6}=3$			

The optimal production sequence shown in Table 6.8 concurs with Jung et al. (1996). However, Jung et al. (1996) considered two temporary storages to be shared commonly between stages and named it as common intermediate storage (CIS). Whereas in the present study, it is assumed that there is a dedicated storage available between every two stages. The results show the location and amount of inter-stage idle time, waiting time inside the temporary storage and holding time within the same stage. A Gantt chart is shown in Figure 6.6 to illustrate the optimal production sequence shown in Table 6.8.

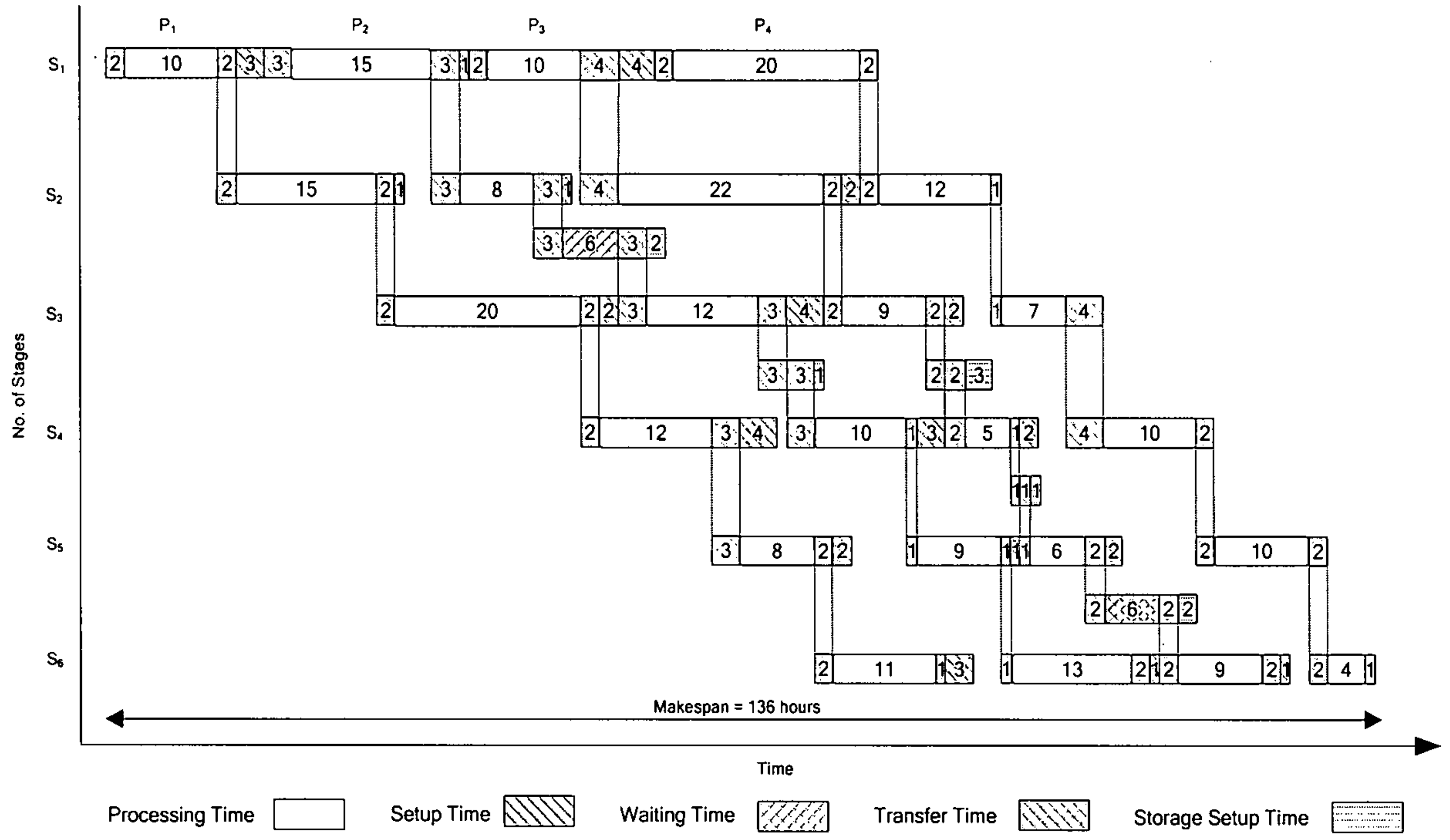


Figure 6.6 Gantt chart for production sequence $P_1P_2P_3P_4$ for case study 2 (FIS/UW transfer policy)

6.4.3 Case Study 3 (Kim et al., 1996)

The batch process recipe for producing eight products in four stages is given in Table 6.9. The corresponding stage transfer time and stage setup time are also given in the same table.

Table 6.9. Batch process recipe for eight products and four stages for case study 3

	Stage Processing Time				Stage Transfer Time				
Product	S_1	S_2	S_3	S_4	T_0	T_1	T_2	T_3	T_4
P_1	10	20	5	30	2	2	2	2	3
P_2	15	8	12	10	3	3	3	3	1
P_3	20	7	9	5	2	4	2	2	1
P_4	13	7	17	10	2	2	1	4	2
P_5	8	3	16	7	1	1	1	3	2
P_6	6	9	22	7	3	3	2	2	2
P_7	7	5	15	12	1	2	1	2	1
P_8	14	13	6	4	2	4	1	1	2

	Stage Setup Time			
Product Sequence	U_1	U_2	U_3	U_4
P_1-P_2	3	1	2	4
P_1-P_3	2	2	1	3
P_1-P_4	1	4	2	2
P_1-P_5	1	3	2	3
P_1-P_6	3	2	1	1
P_1-P_7	3	1	2	1

Product Sequence	Stage Setup Time			
	U_1	U_2	U_3	U_4
P_1-P_8	2	3	3	2
P_2-P_1	4	1	2	3
P_2-P_3	1	1	4	3
P_2-P_4	3	2	3	2
P_2-P_5	3	1	2	2
P_2-P_6	1	1	2	4
P_2-P_7	2	2	1	1
P_2-P_8	3	2	1	3
P_3-P_1	2	1	4	3
P_3-P_2	1	2	3	2
P_3-P_4	2	2	2	2
P_3-P_5	2	1	2	1
P_3-P_6	2	1	1	3
P_3-P_7	1	4	2	2
P_3-P_8	2	2	3	1
P_4-P_1	4	3	4	3
P_4-P_2	1	4	3	3
P_4-P_3	3	2	2	1
P_4-P_5	1	3	3	2
P_4-P_6	3	1	2	2
P_4-P_7	1	1	3	1
P_4-P_8	2	3	1	3
P_5-P_1	2	2	1	4
P_5-P_2	2	2	3	2
P_5-P_3	2	3	3	3

Product Sequence	Stage Setup Time			
	U_1	U_2	U_3	U_4
P_5-P_4	1	1	3	4
P_5-P_6	4	3	3	1
P_5-P_7	2	3	2	3
P_5-P_8	4	3	2	2
P_6-P_1	1	2	1	2
P_6-P_2	3	3	2	3
P_6-P_3	4	1	1	2
P_6-P_4	2	3	4	1
P_6-P_5	2	1	1	4
P_6-P_7	2	2	3	2
P_6-P_8	1	3	3	3
P_7-P_1	2	3	2	1
P_7-P_2	1	1	1	1
P_7-P_3	2	1	2	2
P_7-P_4	1	2	1	3
P_7-P_5	3	2	2	2
P_7-P_6	1	2	4	1
P_7-P_8	2	1	1	1
P_8-P_1	1	1	2	1
P_8-P_2	4	1	4	4
P_8-P_3	2	1	3	2
P_8-P_4	3	4	1	1
P_8-P_5	3	1	3	2
P_8-P_6	3	3	1	1
P_8-P_7	2	3	1	1

A summary of results for optimal sequences obtained in case of ZW, NIS/UW and UIS/UW transfer policies is shown in Table 6.10.

Table 6.10: Summary of results for optimal production sequence for case study 3

Transfer Policy	Optimal Production Sequence	Makespan (hour)	Slack Variables (hour)			CPU Time (sec)
			Inter-Stage Idle Time + Stage Setup Time (H)	Holding Time (I)	Waiting Time (W)	
ZW	$P_5P_1P_6P_3P_7$ $P_4P_2P_8$	195	$H_{1,1}=2$			4.396
			$H_{1,2}=10$			
			$H_{1,3}=13$			
			$H_{1,4}=11$			
			$H_{2,1}=18$			
			$H_{2,2}=5$			
			$H_{2,3}=10$			
			$H_{2,4}=1$			
			$H_{3,1}=4$			
			$H_{3,2}=15$			
			$H_{3,3}=2$			
			$H_{3,4}=4$			
			$H_{4,1}=7$			
			$H_{4,2}=6$			
			$H_{4,3}=2$			
			$H_{4,4}=12$			
			$H_{5,1}=1$			
			$H_{5,2}=10$			
			$H_{5,3}=2$			
			$H_{5,4}=7$			
			$H_{6,1}=3$			
			$H_{6,2}=13$			
			$H_{6,3}=3$			

			$H_{6,4}=6$			
			$H_{7,1}=3$			
			$H_{7,2}=8$			
			$H_{7,3}=10$			
			$H_{7,4}=6$			
NIS/UW	$P_5P_7P_2P_4P_1$	185	$H_{1,1}=2$	$I_{1,1}=0$		4.366
	$P_6P_8P_3$		$H_{1,2}=6$	$I_{1,2}=8$		
			$H_{1,3}=2$	$I_{1,3}=0$		
			$H_{1,4}=9$	$I_{2,1}=0$		
			$H_{2,1}=1$	$I_{2,2}=2$		
			$H_{2,2}=5$	$I_{2,3}=0$		
			$H_{2,3}=1$	$I_{3,1}=0$		
			$H_{2,4}=3$	$I_{3,2}=4$		
			$H_{3,1}=3$	$I_{3,3}=0$		
			$H_{3,2}=5$	$I_{4,1}=0$		
			$H_{3,3}=3$	$I_{4,2}=0$		
			$H_{3,4}=10$	$I_{4,3}=3$		
			$H_{4,1}=4$	$I_{5,1}=12$		
			$H_{4,2}=4$	$I_{5,2}=0$		
			$H_{4,3}=5$	$I_{5,3}=6$		
			$H_{4,4}=3$	$I_{6,1}=0$		
			$H_{5,1}=3$	$I_{6,2}=10$		
			$H_{5,2}=2$	$I_{6,3}=2$		
			$H_{5,3}=4$	$I_{7,1}=1$		
			$H_{5,4}=1$	$I_{7,2}=0$		
			$H_{6,1}=1$	$I_{7,3}=0$		
			$H_{6,2}=6$			
			$H_{6,3}=3$			
			$H_{6,4}=3$			
			$H_{7,1}=2$			
			$H_{7,2}=1$			
			$H_{7,3}=3$			
			$H_{7,4}=8$			
UIS/UW	$P_5P_4P_1P_7P_2$	173	$H_{1,1}=1$		$W_{1,1}=0$	4.367
	$P_6P_3P_8$		$H_{1,2}=12$		$W_{1,2}$=pass	
			$H_{1,3}=3$		$W_{1,3}=0$	

$H_{1,4}=12$	$W_{2,1}=0$
$H_{2,1}=4$	$W_{2,2}=0$
$H_{2,2}=8$	$W_{2,3}=0$
$H_{2,3}=8$	$W_{3,1}=10$
$H_{2,4}=3$	$W_{3,2}$=pass
$H_{3,1}=3$	$W_{3,3}=14$
$H_{3,2}=1$	$W_{4,1}=0$
$H_{3,3}=2$	$W_{4,2}=4$
$H_{3,4}=1$	$W_{4,3}=11$
$H_{4,1}=1$	$W_{5,1}$=pass
$H_{4,2}=1$	$W_{5,2}=8$
$H_{4,3}=1$	$W_{5,3}=1$
$H_{4,4}=1$	$W_{6,1}=0$
$H_{5,1}=1$	$W_{6,2}=10$
$H_{5,2}=2$	$W_{6,3}$=pass
$H_{5,3}=2$	$W_{7,1}=0$
$H_{5,4}=4$	$W_{7,2}=0$
$H_{6,1}=4$	$W_{7,3}=0$
$H_{6,2}=12$	
$H_{6,3}=1$	
$H_{6,4}=2$	
$H_{7,1}=2$	
$H_{7,2}=9$	
$H_{7,3}=3$	
$H_{7,4}=2$	

It is observed from Table 6.10 that the proposed method appears to be efficient and requires reasonable time to generate the optimal solution. In addition, the proposed method provides similar result as that of Kim et al. (1996) with insights of batch process by determining the amount of idle time between process stages and waiting time inside the temporary storages. The Gantt charts for optimal production sequence(s) obtained in case study 3 are shown in Figure 6.7, 6.8 and 6.9 respectively.

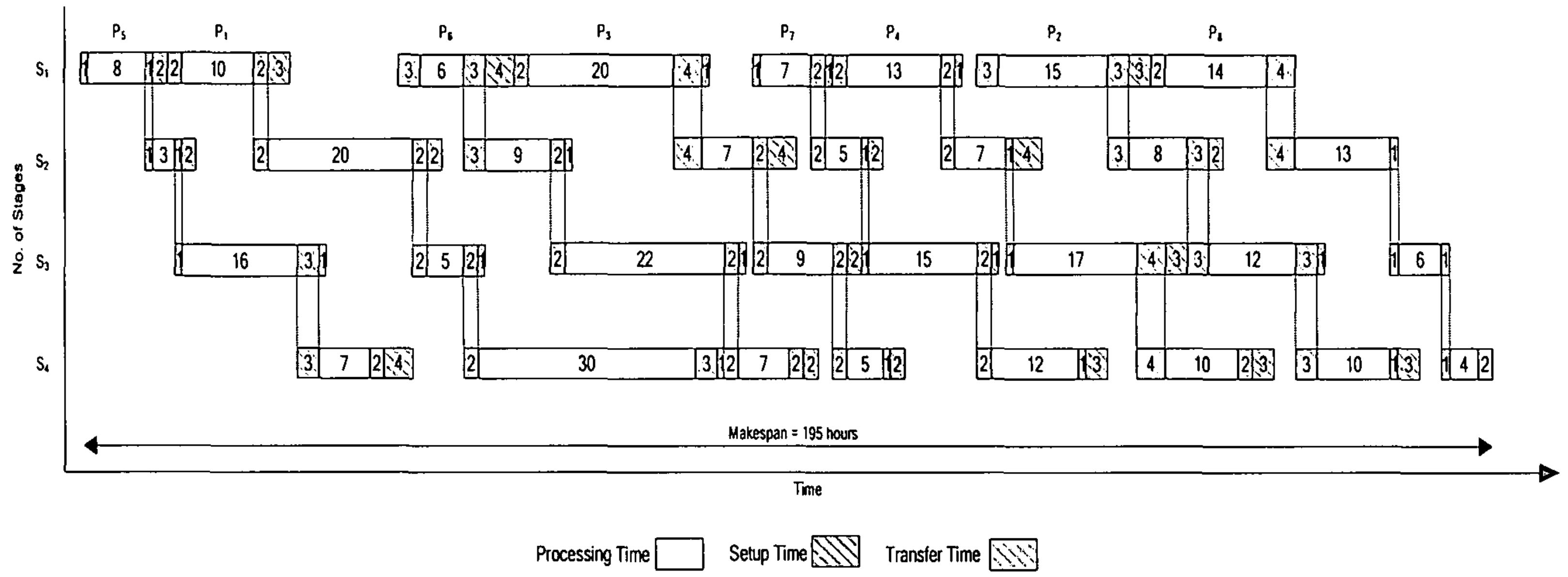


Figure 6.7 Gantt chart for production sequence $P_5P_1P_6P_3P_7P_4P_2P_8$ for case study 3 (ZW transfer policy)

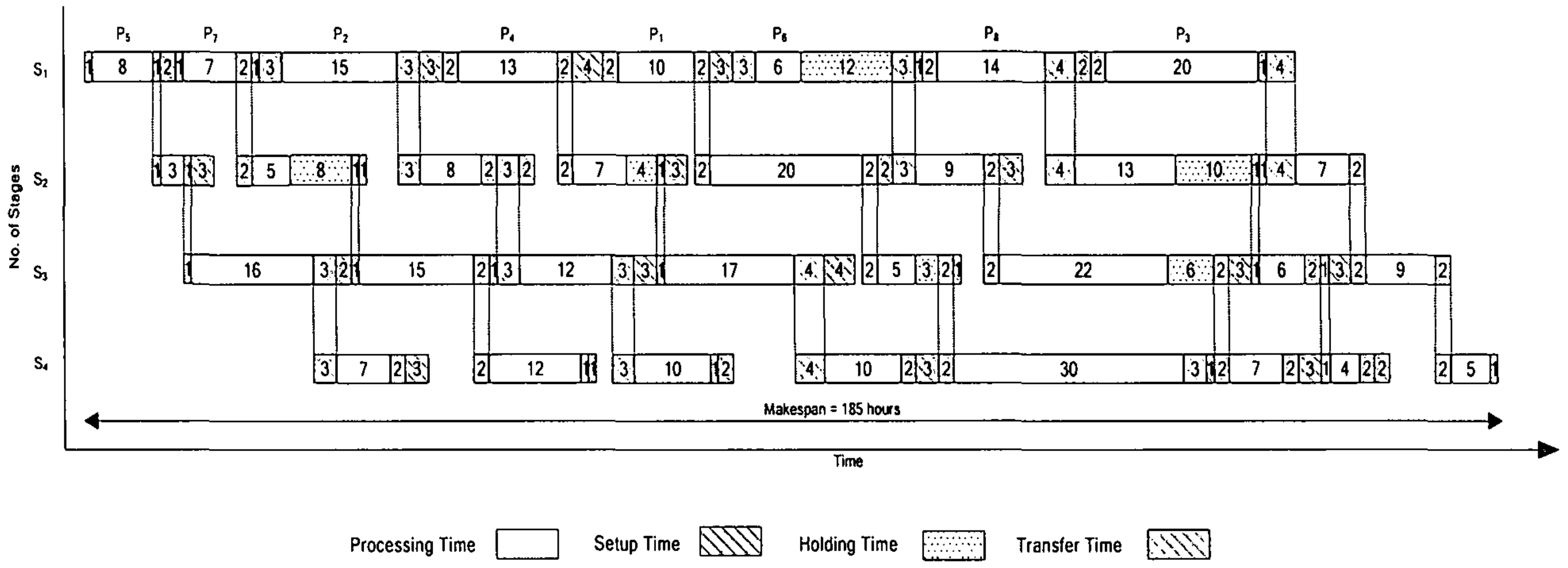


Figure 6.8 Gantt chart for production sequence $P_5P_7P_2P_4P_1P_6P_8P_3$ for case study 3 (NIS/UW transfer policy)

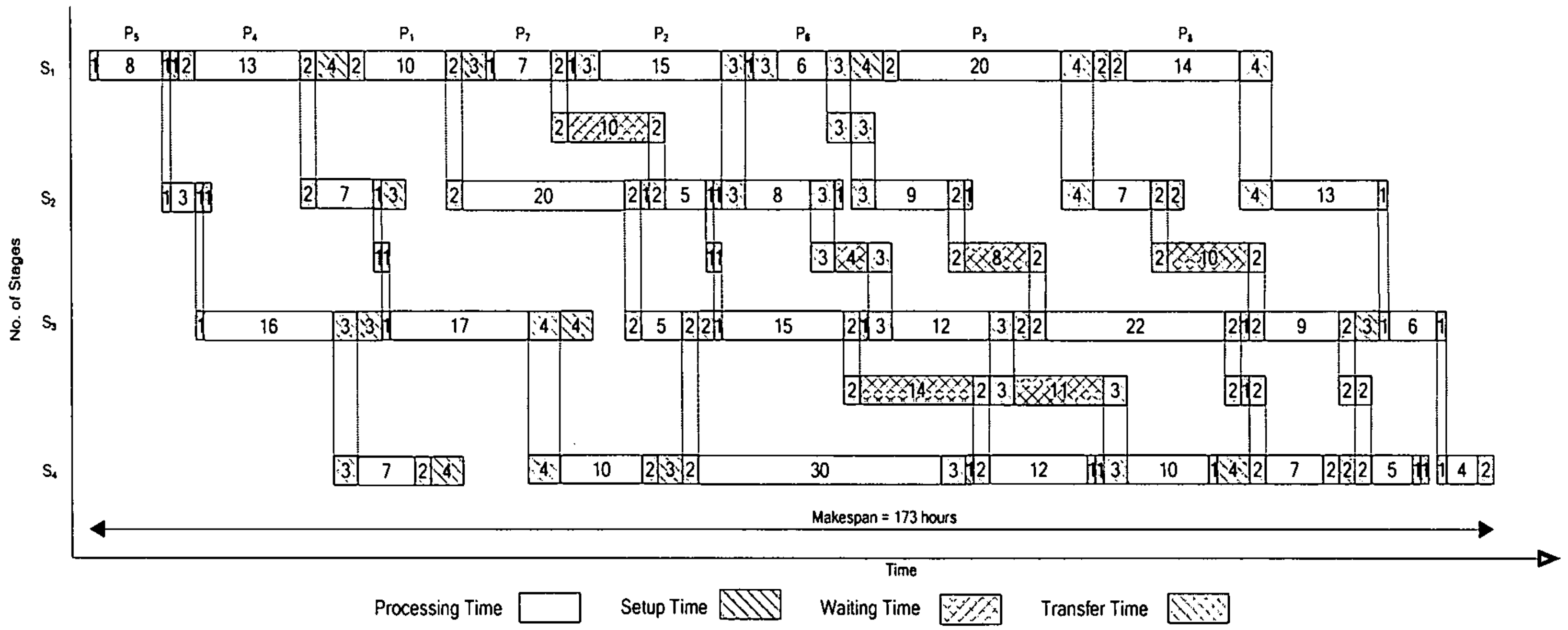


Figure 6.9 Gantt chart for production sequence $P_5P_4P_1P_7P_2P_6P_3P_8$ for case study 3 (UIS/UW transfer policy)

6.5 Summary

Among the purposes of the proposed method is to also provide process planner with a set of options from which he/she could choose for the best solution. These options can be generated by applying the proposed method on all possible production sequences of the given batch process recipe. Different production sequences produce different results for makespan and inter-stage idle time with respect to the transfer policy adopted. Based on these results, the process planner can select the feasible solution according to some predefined criteria. The criteria may either be minimum makespan, inter-stage idle time or in particular the number of temporary storages required. The final decision on the feasible solution could be made if such information is provided for different possible production sequences of the given batch process recipe. However, the computational time increases with problem size. This is because all possible solutions are searched before ending up with the optimal solution. To reduce the solution search space, a set of heuristic rules have been proposed and also shown to reduce the computational time significantly for various problem sizes. Lastly, the application of the proposed procedure has been shown using some case studies available from the relevant literature for the determination of optimal solution.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

The smooth operation of a batch process can be carried out by scheduling the various tasks involved in the production using any one or mix of the intermediate transfer policies. Among others, the most common objective of scheduling is to minimize the makespan of the batch process within a reasonable computational time. The makespan can be minimized in a number of ways such as by proper sequencing i.e. the order of producing the various products on the same batch facility and by using additional stages in parallel to the existing ones for simultaneous unit operation. The batch scheduling problems can be solved for minimum makespan using the available methods developed for this purpose. These methods involve various linear and non linear programming techniques such as MILP or MINLP as well as search techniques such as simulated annealing, genetic algorithm and tabu search. The optimal solution for a batch process could be the one that offers a production sequence with minimum makespan. The newly developed method in this work using matrix representation is suggested as an alternate to the available methods for determining the makespan of any batch process operated under different intermediate transfer policies and also offers optimal solution with less computational effort. The method is based on the study of a number of batch process examples using the Gantt chart.

For the purpose of development of new method in the present work, firstly, the transfer and setup times were assumed negligible for the sake of simplicity. The example batch process recipes were analyzed using Gantt charts and afterwards represented in the form of a matrix. The matrix was then solved using a step by step procedure to determine the makespan of a sequence of products arranged in the matrix for each of the transfer and storage policies taken from the literature. It was observed from the results obtained that in addition to the makespan determination, the developed method was also capable of giving some insights that would be very useful for batch scheduling. The insights refer to other information such as idle time between process stages, holding time of product intermediates within the same stage and number of temporary storages required with respect to the transfer policy adopted. Later, some modifications in the developed methods were suggested to incorporate the sequence dependent transfer and setup time as proposed in the literature. As expected, it was observed that with the inclusion of transfer and setup time, the makespan of the respective production sequence increased. For the determination of optimal production sequence, the repetitive application of the developed method was done on all other possible sequences of the products to determine the makespan of each. The results, so obtained, were analyzed to find the optimal production sequence that yielded minimum makespan.

Lastly, a set of heuristic rules have been proposed to reduce the solution search space using partial enumeration. It has been shown that CPU time is reduced significantly compared with total enumeration used earlier for the determination of optimal solution. For comparison purpose, various problem sizes have been solved using both complete and partial enumeration. For verification purpose, case studies from the relevant literature were used to demonstrate the effectiveness of the newly developed method. In addition, the developed method was also found to give promising results within a reasonable computational time as observed in the simulation studies conducted for various problem sizes.

For speedy calculation, a computer code was developed using Microsoft Visual C++ ^{TM} tool. The features of the developed computer code include its user friendly nature and its ability to work with any batch process recipe. The user has only to input the batch process recipe data and the choice of transfer policy on the basis of which the useful information including makespan for the given batch process recipe are generated.

7.2 Future Work

The scope of the present work is confined to the development of an alternate method to determine the makespan for a multiproduct batch process for ZW and UW transfer policies with various storage configurations and to find the optimal solution with less computational effort compared with mathematical methods available in the past literature.

Further extension of the proposed method could be done to develop the makespan algorithms for other transfer policies stated earlier i.e. LW and MW. In addition, the scheduling of multipurpose batch processes is also as important as multiproduct batch processes. The proposed matrix approach could be explored for its application for multipurpose batch processes. Batch scheduling under uncertainty caused due to unforeseen events could also be the potential research areas identified in the literature. The unforeseen events could be due to late raw material delivery and frequent changes in the market demand.

Another possible extension of the proposed work is to develop the matrix approach to account for the identical and non identical parallel stages. The parallel stages are usually installed to improve the production capacity and solution performance especially with respect to minimizing the makespan of a batch process.

REFERENCES

1. Azzaro-Pantel, C., Bernal-Haro, L., Baudet, P., Domenech, S. and Pibouleau, L. (1998). A two-stage methodology for short term batch plant scheduling: discrete-event simulation and genetic algorithm. Computers & Chemical Engineering, 22: (10)1461-1481.

2. Balasubramanian, J., and Grossmann, I.E. (2002). A novel branch and bound algorithm for scheduling flowshop plants with uncertain processing times. Computers & Chemical Engineering, 26: (1) 41–57.

3. Barbosa-P'ovoa, A. (2007). A critical review on the design and retrofit of batch plants. Computers & Chemical Engineering, 31: (7) 833-855.

4. Bassett, M. H., Pekny, J. F. and Reklaitis, G. V. (1997). Using detailed scheduling to obtain realistic operating policies for a batch processing facility. Industrial & Engineering Chemistry Research, 36: (5) 1717-1726.

5. Bernal-Haro, L., Azzaro-Pantel, C., Domenech, S. and Pibouleau, L. (1998). Design of multipurpose batch chemical plants using a genetic algorithm. Computers & Chemical Engineering, 22: (1) S777-S780.

6. Biegler, L. T., Grossmann, I. E., and Westberg, A.W. (1997). Systematic methods of chemical process design. Prentice-Hall.

7. Birewar, D.B. and Grossmann, I.E. (1989a). Incorporating scheduling in the optimal design of multiproduct batch plants. Computers & Chemical Engineering, 13: (1-2) 141-161.

8. Birewar, D. B. and Grossmann, I.E. (1989b). Efficient optimization algorithms for zero-wait scheduling of multiproduct batch plants. Industrial & Engineering Chemistry Research, 28: (9) 1333–1345.

9. Birewar, D. B. and Grossmann, I.E. (1990). Simultaneous synthesis, sizing, and scheduling of multiproduct batch plants. Industrial & Engineering Chemistry Research, 29: (11) 2242–2251.

10. Blömer, F. and Günther, H.O. (1998). Scheduling of a multi-product batch process in the chemical industry. Computers in Industry, 36: (3) 245-259.

11. Bonfill, A., Espun, A. and Puigjaner, L. (2005). Addressing robustness in scheduling batch processes with uncertain operation times. Industrial & Engineering Chemistry Research, 44: (5) 1524-1534.

12. Caraffa, V., Ianes, S., Bagchi, T. P. and Sriskandarajah, C. (2001). Minimizing makespan in a blocking flowshop using genetic algorithms. International Journal of Production Economics, 70: (2) 101-115.

13. Castro, P.; Barbosa-Povoa, A. P. F. D. and Matos, H. (2001). An improved RTN continuous-time formulation for the short-term scheduling of multipurpose batch plants. Industrial & Engineering Chemistry Research, 40: (9) 2059-2068.

14. Castro, P. M.; Barbosa-Povoa, A. P. and Novais, A. Q. (2005). Simultaneous design and scheduling of multipurpose plants using resource task network based continuous-time formulations. Industrial & Engineering Chemistry Research, 44: (2) 343-357.

15. Castro, P. M. and Grossmann I.E. (2005). New continuous-time MILP model for the short-term scheduling of multistage batch plants. Industrial & Engineering Chemistry Research, 44: (24) 9175-9190.

16. Cerda, J., Vicente, M., Gutierrez, J. M., Esplugas, S., and Mata, J. (1989). A new methodology for the optimal-design and production schedule of multipurpose batch plants. Industrial & Engineering Chemistry Research, 28: (7) 988–998.

17. Cerda, J., Henning, G. P. and Grossmann, I.E. (1997). A mixed-integer linear programming model for short-term scheduling of single-stage multiproduct batch plants with parallel lines. Industrial & Engineering Chemistry Research, 36: (5) 1695-1707.

18. Chan, K. F. and Hui, C. W. (2003). Scheduling batch production using a stepwise approach. Industrial & Engineering Chemistry Research, 42: (14) 3505-3508.

19. Chen, C.I., Liu, C.I., Feng, X-d. and Shao, H-h. (2002). Optimal short-term scheduling of multiproduct single-stage batch plants with parallel lines. Industrial & Engineering Chemistry Research, 41: (5) 1249-1260.

20. Das, H., Cummings, P.T. and Le Van, M.D. (1990). Scheduling of serial multiproduct batch processes via simulated annealing. Computers & Chemical Engineering, 14: (12) 1351-1362.

21. Edgar, T.F., Himmelblau, D. M. and Lasdon, L.S. (2001). Optimization of chemical processes. 2nd Edition, McGraw Hill.

22. Egli, U.M. and Rippin, D.W.T. (1986). Short-term scheduling for multiproduct batch chemical plants. Computers & Chemical Engineering, 10: (4) 303-325.

23. Faqir, N. M. and Karimi, I.A. (1989). Optimal-design of batch plants with single production routes. Industrial & Engineering Chemistry Research, 28: (8) 1191–1202.

24. Giannelos, N. F. and Georgiadis, M. C. (2002a). A simple new continuous-time formulation for short-term scheduling of multipurpose batch processes. Industrial & Engineering Chemistry Research, 41: (9) 2178-2184.

25. Giannelos, N. F. and Georgiadis, M. C. (2002b). A novel event-driven formulation for short-term scheduling of multipurpose continuous processes. Industrial & Engineering Chemistry Research, 41: (10) 2431-2439.

26. Graells, M., Cuxart, J., Espuna, A. and Puigjaner, L. (1995). Dispatching-like strategies using intermediate storage for the scheduling of batch chemical processes. Computers & Chemical Engineering, 19: (1) S621-S626.

27. Graells, M., Espuna, A. and Puigjaner, L. (1996). Sequencing intermediate products: A practical solution for multipurpose production scheduling. Computers & Chemical Engineering, 20: (2) S1137-S1142.

28. Grau, R., Espuna, A. and Puigjaner, L. (1996). Completion times in multipurpose batch plants with set-up, transfer and clean-up times. Computers & Chemical Engineering, 20: (2) S1143-S1148.

29. Gupta, S. and Karimi, I.A. (2003a). Scheduling a two stage multiproduct process with limited product shelf life in Intermediate storage. Industrial & Engineering Chemistry Research, 42: (3) 490-508.

30. Gupta, S. and Karimi, I.A. (2003b). An improved MILP formulation for scheduling multiproduct, multistage batch plants. Industrial & Engineering Chemistry Research, 42: (11) 2365-2380.

31. Ha, J.K., Chang, H.K., Lee, E. S., Lee, I.B., Lee, B.S. and Yi, G. (2000). Intermediate storage tank operation strategies in the production scheduling of multi-product batch processes. Computers & Chemical Engineering, 24: (2-7) 1633-1640.

32. He, Y. and Hui, C.W. (2006). Rule-Evolutionary approach for single-stage multiproduct scheduling with parallel units. Industrial & Engineering Chemistry Research, 45: (13) 4679-4692.

33. Heo, S.K., Lee, K.H., Lee, H.K. Lee, I.B. and Park, J.H. (2003). A new algorithm for cyclic scheduling and design of multipurpose batch plants. Industrial & Engineering Chemistry Research, 42: (4) 836-846.

34. Henning, G. P., Camussi, N. B. and Cerda, J. (1994). Design and planning of multipurpose plants involving nonlinear processing networks. Computers & Chemical Engineering, 18: (2)129–152.

35. Hui, C. W. and Gupta, A. (2001). A bi-index continuous-time mixed-integer linear programming model for single-stage batch scheduling with parallel units. Industrial & Engineering Chemistry Research, 40: (25) 5960-5967.

36. Janicke, W. (1984). On the solution of scheduling problems for multi-purpose batch chemical plants . Computers & Chemical Engineering, 8: (6) 339-343.

37. Janicke, W. (1987). On the design of multipurpose batch chemical plants. Computers in Industry, 9: (1) 19–24.

38. Jung, J.H., Lee, H.K., Yang, D. and Lee, I.B. (1994). Completion times and optimal scheduling for serial multi-product processes with transfer and set-up times in zero-wait policy. Computers & Chemical Engineering, 18: (6) 537–543.

39. Jung, J. H., Lee, H. K. and Lee, I. B. (1996). Completion times algorithm of multi-product batch processes for common intermediate storage policy (CIS) with nonzero transfer and set-up times. Computers & Chemical Engineering, 20: (6-7) 845-852.

40. Karimi, I.A. and Reklaitis, G.V. (1983). Optimal selection of intermediate storage capacity in a periodic batch / semi continuous process. American Institute of Chemical Engineers Journal, 29: (4) 588-596.

41. Karimi, I.A. and Ku, H. M. (1988). A modified heuristic for an initial sequence in flowshop scheduling. Industrial & Engineering Chemistry Research, 27: (9) 1654-1658.

42. Király, L. M., Friedler, F. and Szoboszlai, L. (1989). Optimal design of multi-purpose batch chemical plants. Computers & Chemical Engineering, 13: (4-5) 527–534.

43. Knopf, C.F. (1985). Sequencing a generalized two-stage flowshop with finite intermediate storage. Computers & Chemical Engineering, 9: (3) 207-221.

44. Kim, M., Jung, J.H. and Lee, I.B. (1996). Optimal scheduling of multiproduct batch processes for various intermediate storage policies. Industrial & Engineering Chemistry Research, 35: (11) 4058-4066.

45. Kim, S. B., Lee, H.K., Lee, I.B., Lee, E. S. and Lee, B. (2000). Scheduling of non-sequential multipurpose batch processes under finite intermediate storage policy. Computers & Chemical Engineering, 24: (2-7) 1603-1610.

46. Kondili, E., Pantelides, C.C. and Sargent, R.W.H. (1993). A general algorithm for short-term scheduling of batch operations—I. MILP formulation. Computers & Chemical Engineering, 17: (2) 211-227.

47. Ku, H.M., Rajagoplan,D., and Karimi, I.A. (1987). Scheduling in batch processes. Chemical Engineering Progress, 83: (8) 35.

48. Ku, H. M. and Karimi, I.A. (1988). Scheduling in serial multiproduct batch processes with finite interstage storage: a mixed integer linear program formulation. Industrial & Engineering Chemistry Research, 27: (10) 1840-1848.

49. Ku, H.M. and Karimi, I.A. (1990). Completion time algorithms for serial multiproduct batch processes with shared storage. Computers & Chemical Engineering, 14 (1), 49-69.

50. Ku, H.M. and Karimi, I.A. (1991a). Scheduling algorithms for serial multiproduct batch processes with tardiness penalties. Computers & Chemical Engineering, 15: (5) 283-286.

51. Ku, H. M. and Karimi, I.A. (1991b). An evaluation of simulated annealing for batch process scheduling. Industrial & Engineering Chemistry Research, 30: (1) 163-169.

52. Ku, H.M. and Karimi, I.A. (1992). Multiproduct batch plant scheduling. In CACHE Process Design Case Studies. Grossmann, I. and Morari, M. (Eds.), CACHE Corporation, Austin,TX.

53. Kudva, G., Elkamel, A., Pekny, J.F. and Reklaitis, G.V. (1994). Heuristic algorithm for scheduling batch and semi-continuous plants with production deadlines, intermediate storage limitations and equipment changeover costs. Computers & Chemical Engineering, 18: (9) 859-875.

54. Kuriyan, K., Reklaitis, G.V. and Joglekar, G. (1987). Multiproduct plant scheduling studies using BOSS. Industrial & Engineering Chemistry Research, 26: (8) 1551-1558.

55. Kuriyan, K. and Reklaitis, G.V. (1989). Scheduling network flowshops so as to minimize makespan. Computers & Chemical Engineering, 13: (1-2) 187-200.

56. Lee, H.K. and Lee, I.B. (1996). A synthesis of multiproduct batch plants considering both in-phase and out-of-phase modes. Computers & Chemical Engineering, 20: (1) S195-S200.

57. Lee, D. S., Vassiliadis, V. S. and Park, J. M. (2002). List-Based Threshold-Accepting algorithm for zero-wait scheduling of multiproduct batch plants. Industrial & Engineering Chemistry Research, 41: (25) 6579-6588.

58. Lewis, J.P. (1991). Project planning, scheduling and control. Probus Publishing Company. Chicago, Illinois.

59. Liu, Y. and Karimi, I.A. (2007a). Scheduling multistage, multiproduct batch plants with nonidentical parallel units and unlimited intermediate storage. Chemical Engineering Science, 62: (6) 1549-1566.

60. Liu, Y. and Karimi, I.A. (2007b). Novel continuous-time formulations for scheduling multi-stage batch plants with identical parallel units. Computers & Chemical Engineering, 31: (12) 1671-1693.

61. Liu, Y. and Karimi, I.A. (2008). Scheduling multistage batch plants with parallel units and no interstage storage. Computers & Chemical Engineering, 32: (4-5) 671-693.

62. Majozi, T. and Zhu, X. X. (2001). A novel continuous-time MILP formulation for multipurpose batch plants. 1. Short-term scheduling. Industrial & Engineering Chemistry Research, 40: (25) 5935-5949.

63. Maravelias, C. T. and Grossmann, I. E. (2003a). New general continuous-time state-task network formulation for short-term scheduling of multipurpose batch plants. Industrial & Engineering Chemistry Research, 42: (13) 3056-3074.

64. Maravelias, C. T. and Grossmann, I. E. (2003b). Minimization of the makespan with a discrete-time state-task network formulation. Industrial & Engineering Chemistry Research, 42: (24) 6252-6257.

65. Mauderli, A. and Rippin D. W. T. (1979). Production planning and scheduling for multi-purpose batch chemical plants. Computers & Chemical Engineering, 3: (1-4) 199-206.

66. Mendez, C.A., Cerda, J., Grossmann, I. E., Harjunkoski, I. and Fahl, M. (2006). State-of-the-art review of optimization methods for short-term scheduling of batch processes. Computers & Chemical Engineering, 30: (6-7) 913–946.

67. Modi, A.K. and Karimi, I.A. (1989). Design of multiproduct batch processes with finite intermediate storage. Computers & Chemical Engineering, 13: (1-2) 127-139.

68. Moon, S., Park, S. and Lee, W. K. (1996). New MILP models for scheduling of multiproduct batch plants under zero-wait policy. Industrial & Engineering Chemistry Research, 35: (10) 3458-3469

69. Moon, S. and Hrymak, A. N. (1999). Mixed-Integer linear programming model for short-term scheduling of a special class of multipurpose batch plants. Industrial & Engineering Chemistry Research, 38: (5) 2144-2150.

70. Morrison, S.M. (1996) Scheduling methods for batch digesters, batch process plants, and shops with the conveyor algorithm. Dissertation University of Texas at Austin.

71. Musier, R.F.H. and Evans, L.B (1989). An approximate method for the production scheduling of industrial batch processes with parallel units. Computers & Chemical Engineering, 13(1-2), 229-238.

72. Orçun, S., Altinel, I.K. and Hortaçsu, Ö. (2001). General continuous time models for production planning and scheduling of batch processing plants: mixed integer linear program formulations and computational Issues. Computers & Chemical Engineering, 25: (2-3) 371-389.

73. Papageorgaki, S. and Reklaitis, G. V. (1990a). Optimal design of multipurpose batch plants. 1. Problem formulation. Industrial & Engineering Chemistry Research, 29: (10) 2054–2062.

74. Papageorgaki, S. and Reklaitis, G. V. (1990b). Optimal design of multipurpose batch plants. 2. A decomposition solution strategy. Industrial & Engineering Chemistry Research, 29: (10) 2062–2073.

75. Papageorgaki, S. and Reklaitis, G. V. (1993). Retrofitting a general multipurpose batch chemical-plant. Industrial & Engineering Chemistry Research, 32: (2) 345–362.

76. Patel, A.N., Mah, R.S.H. and Karimi, I.A. (1991). Preliminary design of multiproduct noncontinuous plants using simulated annealing. Computers & Chemical Engineering, 15: (7) 451-469.

77. Patsidou, E.P. and Kantor, J.C. (1991). Scheduling of a multipurpose batch plant using a graphically derived mixed-integer linear program model. Industrial & Engineering Chemistry Research, 30: (7) 1548-1561.

78. Petkov, S. B. and Maranas, C. D. (1997). Multiperiod planning and scheduling of multiproduct batch plants under demand uncertainty. Industrial & Engineering Chemistry Research, 36 (11) 4864-4881.

79. Petkov, S. B. and Maranas, C. D. (1998a). Design of single-product campaign batch plants under demand uncertainty. American Institute of Chemical Engineers Journal, 44: (4) 896–911.

80. Petkov, S. B. and Maranas, C. D. (1998b). Design of multiproduct batch plants under demand uncertainty with staged capacity expansions. Computers & Chemical Engineering, 22: (1) S789–S792.

81. Pinto, J.M. and Grossmann, I.E. (1995). A continuous time mixed integer linear programming model for short term scheduling of multistage batch plants. Industrial & Engineering Chemistry Research, 34: (9) 3037-3051.

82. Pitty, S.S. and Karimi, I.A. (2008). Novel MILP models for scheduling permutation flowshops. Chemical Product and Process Modeling, 3: (1) 1-46.

83. Pozivil, J. and Zdansky, M. (2001). Application of genetic algorithms to chemical flowshop sequencing. Chemical Engineering and Technology, 24: (4) 327-333.

84. Rajagopalan, D. and Karimi, I.A. (1989). Completion times in serial mixed-storage multiproduct processes with transfer and set-up times. Computers & Chemical Engineering, 13: (1-2)175-186.

85. Ravemark, D.E. and Rippin, D.W. T. (1998). Optimal design of a multi-product batch plant. Computers & Chemical Engineering, 22: (1-2) 177–183.

86. Rippin, D. W. T. (1983a). Design and operation of multiproduct and multipurpose batch chemical plants. — An analysis of problem structure. Computers & Chemical Engineering, 7: (4) 463–481.

87. Rippin, D.W. T. (1983b). Simulation of single and multiproduct batch chemical plants for optimal design and operation. Computers & Chemical Engineering, 7: (3) 137–156.

88. Rodrigues, M.T.M., Latre, L. G. and Rodrigues, L. C.A. (2000a). Short-term planning and scheduling in multipurpose batch chemical plants: a multi-level approach. Computers & Chemical Engineering, 24: (9-10) 2247–2258.

89. Rodrigues, M. T. M., Latre, L. G. and Rodrigues, L. C. A. (2000b). Production planning using time windows for short-term multipurpose batch plants scheduling problems. Industrial & Engineering Chemistry Research, 39: (10) 3823-3834.

90. Romero, J., Puigjaner, L., Holczinger, T. and Friedler, F. (2004). Scheduling intermediate storage multipurpose batch plants using the S-graph. American Institute of Chemical Engineers, 50: (2) 403-417.

91. Ryu, J.H. and Lee, I.B. (1997). A new completion time algorithm considering an out-of-phase policy in batch processes. Industrial & Engineering Chemistry Research, 36: (12) 5321-5328.

92. Ryu, J.H., Lee, H.K. and Lee, I.B. (2001). Optimal scheduling for a multiproduct batch process with minimization of penalty on due date period. Industrial & Engineering Chemistry Research, 40: (1) 228-233.

93. Ryu, J.H.and Pistikopoulos, E.N. (2007). A novel approach to scheduling of zero-wait batch processes under processing time variations. Computers & Chemical Engineering, 31: (3) 101–106.

94. Ryu, J., Dua, V. and Pistikopoulos, E. N. (2007). Proactive scheduling under uncertainty: A parametric optimization approach. Industrial & Engineering Chemistry Research, 46(24); 8044-8049.

95. Sanmarti,E., Friedler, F. and Puigjaner, L. (1998). Combinatorial technique for short term scheduling of multipurpose batch plants based on schedule-graph representation. Computers & Chemical Engineering, 22: (1) S847-S850.

96. Smith, R. (2005). Chemical process design and integration. John Wiley & Sons.

97. Suhami, I. and Mah, R.S.H. (1981). An implicit enumeration scheme for the flowshop problem with no intermediate storage. Computers & Chemical Engineering, 5: (2) 83-91.

98. Suhami, I. and Mah, R.S.H. (1982). Optimal design of multipurpose batch plants. Industrial & Engineering Chemistry Research, Process Design and Development, 21: (1) 94-100.

99. Takeichiro T., Iori H. and Shinji H. (1982). Optimal design and operation of a batch process with intermediate storage tanks. Industrial & Engineering Chemistry Research, 21: (3) 431-440.

100. Takeichiro T., Iori H., Shinji H. and Masahiro O.S. (1984). Design of a flexible batch process with intermediate storage tanks. Industrial & Engineering Chemistry Research, 23: (1) 40-48.

101. Tra N.T.L. (2000). Comparison of scheduling algorithms for a multi-product batch-chemical plant with a generalized serial network. Master's thesis, Virginia Polytechnic Institute and State University.

102. Vaselenak, J.A., Grossmann, I.E. and Westerberg, A.W. (1987). An embedding formulation for the optimal scheduling and design of multipurpose batch plants. Industrial & Engineering Chemistry Research, 26: (1)139-148.

103. Vecchietti, A.R. and Montagna, J. (1998). Alternatives in the optimal allocation of intermediate storage tank in multiproduct batch plants. Computers & Chemical Engineering, 22: (1) S801-S804.

104. Vin, J. P. and Ierapetritou, M. G. (2001). Robust short-term scheduling of multiproduct batch plants under demand uncertainty. Industrial & Engineering Chemistry Research, 40: (21) 4543-4554.

105. Vin, J. P.and Ierapetritou, M. G. (2000). A new approach for efficient rescheduling of multiproduct batch plants. Industrial & Engineering Chemistry Research, 39: (11) 4228-4238.

106. Voudouris, V.T. and Grossmann, I.E. (1992). Mixed-integer linear programming reformulations for batch process design with discrete equipment sizes. Industrial & Engineering Chemistry Research, 31: (5)1315-1325.

107. Voudouris, V.T. and Grossmann, I.E. (1993). Optimal synthesis of multiproduct batch plants with cyclic scheduling and inventory considerations. Industrial & Engineering Chemistry Research, 32: (9)1962-1980.

108. Voudouris, V.T. and Grossmann, I.E. (1996). MILP model for scheduling and design of a special class of multipurpose batch plants. Computers & Chemical Engineering, 20: (11)1335-1360.

109. Wellons, H.S. and Reklaitis, G.V. (1989). The design of multiproduct batch plants under uncertainty with staged expansion. Computers & Chemical Engineering, 13: (1-2)115-126.

110. Wellons, M.C. and Reklaitis, G.V. (1989a). Optimal schedule generation for a single-product production line—I. Problem formulation. Computers & Chemical Engineering, 13: (1-2)201-212.

111. Wellons, M.C. and Reklaitis, G.V. (1989b). Optimal schedule generation for a single-product production line—II. Identification of dominant unique path sequences. Computers & Chemical Engineering, 13: (1-2)213-227.

112. Wellons, M.C. and Reklaitis, G.V. (1991a). Scheduling of multipurpose batch chemical plants. 1. Formation of single-product campaigns. Industrial & Engineering Chemistry Research, 30: (4) 671-688.

113. Wellons, M.C. and Reklaitis, G.V. (1991b). Scheduling of multipurpose batch chemical plants. 2. Multiple-product campaign formation and production planning. Industrial & Engineering Chemistry Research, 30: (4) 688-705.

114. Wiede, W., Kuriyan, K. and Reklaitis, G.V. (1987). Determination of completion times for serial multiproduct processes—1. A two unit finite intermediate storage system. Computers & Chemical Engineering, 11: (4) 337-344.

115. Wiede, W. and Reklaitis, G.V. (1987). Determination of completion times for serial multiproduct processes—3. Mixed intermediate storage systems. Computers & Chemical Engineering, 11: (4) 357-368.

116. Zhu, X. X. and Majozi, T. (2001). Novel Continuous Time MILP Formulation for Multipurpose Batch Plants. 2. Integrated Planning and Scheduling. Industrial & Engineering Chemistry Research, 40: (23) 5621-5634.

PUBLICATIONS

Journal Publications:

1. Amir Shafeeq, M.I. Abdul Mutalib, K.A. Amminudin and Ayyaz Muhammad, "Determination of inter-stage idle times in scheduling design of a three stage zero wait batch process". *Journal of Applied Sciences 8(9):1706-1712,2008.*

2. Amir Shafeeq, M.I. Abdul Mutalib, K.A. Amminudin and Ayyaz Muhammad, "Scheduling three stage flowshop processes with no intermediate storage using novel matrix approach". *Journal of Applied Sciences 8(11):2136-2141,2008.*

3. Amir Shafeeq, M.I. Abdul Mutalib, K.A. Amminudin and Ayyaz Muhammad, "New completion time algorithms for sequence based scheduling in multiproduct batch processes using matrix". *Chemical Engineering Research and Design (accepted for publication).*

4. Amir Shafeeq, M. I. Abdul Mutalib, K. A. Amminudin and Ayyaz Muhammad, "More on completion time algorithms for intermediate storage tanks in multiproduct batch process scheduling using matrix representation". *Industrial and Engineering Chemistry Research (accepted for publication).*

Conference Proceedings:

1. Amir Shafeeq, M.I. Abdul Mutalib, K.A. Amminudin and Ayyaz Muhammad, "Determination of makespan for zero wait batch processes: a novel method using matrix". *Proceedings of International MultiConference of Engineers and Computer Scientists, Hongkong, Vol.(II): 21-23 March, 2007, pp 2038-2042.*

2. Amir Shafeeq, M.I. Abdul Mutalib, K.A. Amminudin and Ayyaz Muhammad, "Completion time in single product chemical batch process with parallel unit for ZW/NIS/UIS transfer policies". *Proceedings of 21st Symposium of Malaysian Chemical Engineers, Malaysia, 12-14 December, 2007.*

3. Amir Shafeeq, M.I. Abdul Mutalib, K.A. Amminudin and Ayyaz Muhammad, "A heuristic method to search for optimal solution using partial enumeration for a multiproduct chemical batch process". *Proceedings of 22nd Symposium of Malaysian Chemical Engineers, Malaysia, 2-3 December, 2008. (abstract accepted)*

Book Chapter:

1. Amir Shafeeq, M.I. Abdul Mutalib, K.A. Amminudin and Ayyaz Muhammad, "A novel matrix approach to determine makespan for zero wait batch processes". *Advances in industrial engineering and operations research published by Springer, 2008, pp 143-153.*

APPENDIX A

Sample Calculation for Matrix Approach using problem size of (10 x 5):

The product recipe (Products=10, Stages 5) for the sample calculation has been given below with sequence dependent transfer and setup times. The sample calculation has been performed on two of the important and most applied transfer policies in real world applications (Pitty and Karimi, 2008) i.e. ZW and NIS/UW. The purpose of the sample calculation is to show that the matrix approach could be applied on any problem size and transfer policies discussed in this work to determine the makespan. In addition, for illustration purpose, Gantt charts are also drawn at the end of these calculations to further confer the results obtained using matrix approach.

Table. Batch process recipe for ten products and five stages

	Stage Processing Time					Stage Transfer Time							Stage Setup Time				
Product	S_1	S_2	S_3	S_4	S_5	T_0	T_1	T_2	T_3	T_4	T_5	Product Sequence	U_1	U_2	U_3	U_4	U_5
P_1	84	36	164	70	79	1	1	1	3	2	2	P_1-P_2	2	2	1	4	4
P_2	110	215	55	324	60	2	2	2	2	3	2	P_2-P_3	3	2	1	1	2
P_3	68	96	220	61	56	3	3	2	2	2	2	P_3-P_4	4	1	1	2	2
P_4	218	72	98	52	60	2	4	2	2	1	2	P_4-P_5	1	4	2	2	2
P_5	71	53	155	129	36	1	2	1	2	1	2	P_5-P_6	1	2	1	3	2
P_6	134	77	172	111	68	2	2	1	4	2	2	P_6-P_7	1	4	3	3	2
P_7	153	85	127	113	40	3	3	3	3	1	2	P_7-P_8	3	2	1	3	3
P_8	144	131	62	41	54	2	4	1	1	2	2	P_8-P_9	3	3	3	3	3
P_9	157	135	58	45	50	2	4	1	1	2	3	P_9-P_{10}	3	2	2	2	2
P_{10}	142	129	61	30	24	2	4	1	1	3	3						

ZW Transfer Policy

The batch process recipe is represented to form a matrix $M_{i,j}$ to illustrate the arrangement of variables 'V' that would determine the idle time between stages.

			j		
i	1	2	3	4	5
1	84	36	164	70	79
	$V_{1,1}$	$V_{1,2}$	$V_{1,3}$	$V_{1,4}$	$V_{1,5}$
2	110	215	55	324	60
	$V_{2,1}$	$V_{2,2}$	$V_{2,3}$	$V_{2,4}$	$V_{2,5}$
3	68	96	220	61	56
	$V_{3,1}$	$V_{3,2}$	$V_{3,3}$	$V_{3,4}$	$V_{3,5}$
4	218	72	98	52	60
	$V_{4,1}$	$V_{4,2}$	$V_{4,3}$	$V_{4,4}$	$V_{4,5}$
5	71	53	155	129	36
	$V_{5,1}$	$V_{5,2}$	$V_{5,3}$	$V_{5,4}$	$V_{5,5}$
6	134	77	172	111	68
	$V_{6,1}$	$V_{6,2}$	$V_{6,3}$	$V_{6,4}$	$V_{6,5}$
7	153	85	127	113	40
	$V_{7,1}$	$V_{7,2}$	$V_{7,3}$	$V_{7,4}$	$V_{7,5}$
8	144	131	62	41	54
	$V_{8,1}$	$V_{8,2}$	$V_{8,3}$	$V_{8,4}$	$V_{8,5}$
9	157	135	58	45	50
	$V_{9,1}$	$V_{9,2}$	$V_{9,3}$	$V_{9,4}$	$V_{9,5}$
10	142	129	61	30	24

Following are the equation developed to determine inter-stage idle times with sequence dependent transfer times.

$$V_{i,2} = (M_{i+1,1} + T_{i+1,0}) - (M_{i,2} + T_{i,2}) \qquad i = 1 \ldots\ldots n-1. \qquad (4.1a)$$

$$V_{i,j+1} = (V_{i,j} + M_{i+1,j} + T_{i+1,j-1}) - (M_{i,j+1} + T_{i,j+1}) \qquad j = 2 \ldots\ldots\ldots m-1, \;\; i = 1 \ldots\ldots n-1 \qquad (4.2a)$$

$$V_{i,j} = (V_{i,j+1} - (M_{i+1,j} + T_{i+1,j-1})) + (M_{i,j+1} + T_{i,j+1}) \qquad j = m-1 \ldots\ldots\ldots 1, \;\; i = 1 \ldots\ldots n-1 \qquad (4.3a)$$

Applying the above equations on the sample product recipe gives the following values of the inter-stage idle times.

$V_{1,2}=(M_{2,1}+T_{2,0})-(M_{1,2}+T_{1,2})=75$

$V_{2,2}=(M_{3,1}+T_{3,0})-(M_{2,2}+T_{2,2})=0$

$V_{3,2}=(M_{4,1}+T_{4,0})-(M_{3,2}+T_{3,2})=122$

$V_{4,2}=(M_{5,1}+T_{5,0})-(M_{4,2}+T_{4,2})=0$

$V_{5,2}=(M_{6,1}+T_{6,0})-(M_{5,2}+T_{5,2})=82$

$V_{6,2}=(M_{7,1}+T_{7,0})-(M_{6,2}+T_{6,2})=78$

$V_{7,2}=(M_{8,1}+T_{8,0})-(M_{7,2}+T_{7,2})=58$

$V_{8,2}=(M_{9,1}+T_{9,0})-(M_{8,2}+T_{8,2})=27$

$V_{9,2}=(M_{10,1}+T_{10,0})-(M_{9,2}+T_{9,2})=8$

$V_{1,3}=(V_{1,2}+M_{2,2}+T_{2,1})-(M_{1,3}+T_{1,3})=125$

$V_{1,4}=(V_{1,3}+M_{2,3}+T_{2,2})-(M_{1,4}+T_{1,4})=110$

$V_{1,5}=(V_{1,4}+M_{2,4}+T_{2,3})-(M_{1,5}+T_{1,5})=355$

$V_{2,3}=(V_{2,2}+M_{3,2}+T_{3,1})-(M_{2,3}+T_{2,3})=42$

$V_{2,4}=(V_{2,3}+M_{3,3}+T_{3,2})-(M_{2,4}+T_{2,4})=0$

$V_{2,5}=(V_{2,4}+M_{3,4}+T_{3,3})-(M_{2,5}+T_{2,5})=1$

$V_{3,3} = (V_{3,2} + M_{4,2} + T_{4,1}) - (M_{3,3} + T_{3,3}) = 0$

$V_{3,4} = (V_{3,3} + M_{4,3} + T_{4,2}) - (M_{3,4} + T_{3,4}) = 37$

$V_{3,5} = (V_{3,4} + M_{4,4} + T_{4,3}) - (M_{3,5} + T_{3,5}) = 33$

$V_{4,3} = (V_{4,2} + M_{5,2} + T_{5,1}) - (M_{4,3} + T_{4,3}) = 0$

$V_{4,4} = (V_{4,3} + M_{5,3} + T_{5,2}) - (M_{4,4} + T_{4,4}) = 103$

$V_{4,5} = (V_{4,4} + M_{5,4} + T_{5,3}) - (M_{4,5} + T_{4,5}) = 172$

$V_{5,3} = (V_{5,2} + M_{5,2} + T_{5,1}) - (M_{5,3} + T_{5,3}) = 4$

$V_{5,4} = (V_{5,3} + M_{5,3} + T_{5,2}) - (M_{5,4} + T_{5,4}) = 47$

$V_{5,5} = (V_{5,4} + M_{5,4} + T_{5,3}) - (M_{5,5} + T_{5,5}) = 124$

$V_{6,3} = (V_{6,2} + M_{6,2} + T_{6,1}) - (M_{6,3} + T_{6,3}) = 0$

$V_{6,4} = (V_{6,3} + M_{6,3} + T_{6,2}) - (M_{6,4} + T_{6,4}) = 17$

$V_{6,5} = (V_{6,4} + M_{6,4} + T_{6,3}) - (M_{6,5} + T_{6,5}) = 63$

$V_{7,3} = (V_{7,2} + M_{7,2} + T_{7,1}) - (M_{7,3} + T_{7,3}) = 63$

$V_{7,4} = (V_{7,3} + M_{7,3} + T_{7,2}) - (M_{7,4} + T_{7,4}) = 12$

$V_{7,5} = (V_{7,4} + M_{7,4} + T_{7,3}) - (M_{7,5} + T_{7,5}) = 12$

$V_{8,3} = (V_{8,2} + M_{8,2} + T_{8,1}) - (M_{8,3} + T_{8,3}) = 103$

$V_{8,4} = (V_{8,3} + M_{8,3} + T_{8,2}) - (M_{8,4} + T_{8,4}) = 119$

$V_{8,5} = (V_{8,4} + M_{8,4} + T_{8,3}) - (M_{8,5} + T_{8,5}) = 109$

$V_{9,3} = (V_{9,2} + M_{10,2} + T_{10,1}) - (M_{9,3} + T_{9,3}) = 82$

$V_{9,4} = (V_{9,3} + M_{10,3} + T_{10,2}) - (M_{9,4} + T_{9,4}) = 97$

$V_{9,5} = (V_{9,4} + M_{10,4} + T_{10,3}) - (M_{9,5} + T_{9,5}) = 75$

Based on the calculated values of $V_{1,5}$ ......$V_{9,5}$, the values of the previous variables 'V' are recalculated using equation (4.3a) to determine the values of $V_{1,1}$......... $V_{9,1}$ as demonstrated below:

$V_{1,4} = (V_{1,5} - (M_{2,4} + T_{2,3})) + (M_{1,5} + T_{1,5}) = 110$

$V_{1,3} = (V_{1,4} - (M_{2,3} + T_{2,2})) + (M_{1,4} + T_{1,4}) = 125$

$V_{1,2} = (V_{1,3} - (M_{2,2} + T_{2,1})) + (M_{1,3} + T_{1,3}) = 75$

$V_{1,1} = (V_{1,2} - (M_{2,1} + T_{2,0})) + (M_{1,2} + T_{1,2}) = 0$

$V_{2,4} = (V_{2,5} - (M_{3,4} + T_{3,3})) + (M_{2,5} + T_{2,5}) = 0$

$V_{2,3} = (V_{2,4} - (M_{3,3} + T_{3,2})) + (M_{2,4} + T_{2,4}) = 105$

$V_{2,2} = (V_{2,3} - (M_{3,2} + T_{3,1}) + (M_{2,3} + T_{2,3}) = 63$

$V_{2,1} = (V_{2,2} - (M_{3,1} + T_{3,0})) + (M_{2,2} + T_{2,2}) = 209$

$V_{3,4} = (V_{3,5} - (M_{4,4} + T_{3,3})) + (M_{3,5} + T_{3,5}) = 37$

$V_{3,3} = (V_{3,4} - (M_{4,3} + T_{3,2})) + (M_{3,4} + T_{3,4}) = 0$

$$V_{3,2} = (V_{3,3} - (M_{4,2} + T_{3,1})) + (M_{3,3} + T_{3,3}) = 146$$

$$V_{3,1} = (V_{3,2} - (M_{4,1} + T_{3,0})) + (M_{3,2} + T_{3,2}) = 24$$

$$V_{4,4} = (V_{4,5} - (M_{4,4} + T_{4,3})) + (M_{4,5} + T_{4,5}) = 103$$

$$V_{4,3} = (V_{4,4} - (M_{4,3} + T_{4,2})) + (M_{4,4} + T_{4,4}) = 0$$

$$V_{4,2} = (V_{4,3} - (M_{4,2} + T_{4,1})) + (M_{4,3} + T_{4,3}) = 45$$

$$V_{4,1} = (V_{4,2} - (M_{4,1} + T_{4,0})) + (M_{4,2} + T_{4,2}) = 47$$

$$V_{5,4} = (V_{5,5} - (M_{5,4} + T_{5,3})) + (M_{5,5} + T_{5,5}) = 47$$

$$V_{5,3} = (V_{5,4} - (M_{5,3} + T_{5,2})) + (M_{5,4} + T_{5,4}) = 4$$

$$V_{5,2} = (V_{5,3} - (M_{5,2} + T_{5,1})) + (M_{5,3} + T_{5,3}) = 82$$

$$V_{5,1} = (V_{5,2} - (M_{5,1} + T_{5,0})) + (M_{5,2} + T_{5,2}) = 0$$

$$V_{6,4} = (V_{6,5} - (M_{6,4} + T_{6,3})) + (M_{6,5} + T_{6,5}) = 17$$

$$V_{6,3} = (V_{6,4} - (M_{6,3} + T_{6,2})) + (M_{6,4} + T_{6,4}) = 0$$

$$V_{6,2} = (V_{6,3} - (M_{6,2} + T_{6,1})) + (M_{6,3} + T_{6,3}) = 88$$

$$V_{6,1} = (V_{6,2} - (M_{6,1} + T_{6,0})) + (M_{6,2} + T_{6,2}) = 10$$

$$V_{7,4} = (V_{7,5} - (M_{7,4} + T_{7,3})) + (M_{7,5} + T_{7,5}) = 12$$

$$V_{7,3} = (V_{7,4} - (M_{7,3} + T_{7,2})) + (M_{7,4} + T_{7,4}) = 63$$

$$V_{7,2} = (V_{7,3} - (M_{7,2} + T_{7,1})) + (M_{7,3} + T_{7,3}) = 58$$

$$V_{7,1} = (V_{7,2} - (M_{7,1} + T_{7,0})) + (M_{7,2} + T_{7,2}) = 0$$

$$V_{8,4} = (V_{8,5} - (M_{8,4} + T_{8,3})) + (M_{8,5} + T_{8,5}) = 119$$

$$V_{8,3} = (V_{8,4} - (M_{8,3} + T_{8,2})) + (M_{8,4} + T_{8,4}) = 103$$

$$V_{8,2} = (V_{8,3} - (M_{8,2} + T_{8,1})) + (M_{8,3} + T_{8,3}) = 27$$

$$V_{8,1} = (V_{8,2} - (M_{8,1} + T_{8,0})) + (M_{8,2} + T_{8,2}) = 0$$

$$V_{9,4} = (V_{9,5} - (M_{9,4} + T_{9,3})) + (M_{9,5} + T_{9,5}) = 97$$

$$V_{9,3} = (V_{9,4} - (M_{9,3} + T_{9,2})) + (M_{9,4} + T_{9,4}) = 82$$

$$V_{9,2} = (V_{9,3} - (M_{9,2} + T_{9,1})) + (M_{9,3} + T_{9,3}) = 8$$

$$V_{9,1} = (V_{9,2} - (M_{9,1} + T_{9,0})) + (M_{9,2} + T_{9,2}) = 0$$

The idle time calculated above are then compared to the required setup times to determine the actual setup times to be used for the purpose of calculating the makespan.

$CT_{1,1}=U_{1,1}-V_{1,1}=2$	$CT_{1,2}=U_{1,2}-V_{1,2}=-73$	$CT_{1,3}=U_{1,3}-V_{1,3}=-124$
$CT_{1,4}=U_{1,4}-V_{1,4}=-106$	$CT_{1,5}=U_{1,5}-V_{1,5}=-351$	$CT_{2,1}=U_{2,1}-V_{2,1}=-206$
$CT_{2,2}=U_{2,2}-V_{2,2}=-61$	$CT_{2,3}=U_{2,3}-V_{2,3}=-104$	$CT_{2,4}=U_{2,4}-V_{2,4}=1$
$CT_{2,5}=U_{2,5}-V_{2,5}=1$	$CT_{3,1}=U_{3,1}-V_{3,1}=-20$	$CT_{3,2}=U_{3,2}-V_{3,2}=-145$
$CT_{3,3}=U_{3,3}-V_{3,3}=1$	$CT_{3,4}=U_{3,4}-V_{3,4}=-35$	$CT_{3,5}=U_{3,5}-V_{3,5}=-31$
$CT_{4,1}=U_{4,1}-V_{4,1}=-46$	$CT_{4,2}=U_{4,2}-V_{4,2}=-41$	$CT_{4,3}=U_{4,3}-V_{4,3}=2$

$CT_{4,4}=U_{4,4}-V_{4,4}=-101$	$CT_{4,5}=U_{4,5}-V_{4,5}=-170$	$CT_{5,1}=U_{5,1}-V_{5,1}=1$
$CT_{5,2}=U_{5,2}-V_{5,2}=-80$	$CT_{5,3}=U_{5,3}-V_{5,3}=-3$	$CT_{5,4}=U_{5,4}-V_{5,4}=-44$
$CT_{5,5}=U_{5,5}-V_{5,5}=-122$	$CT_{6,1}=U_{6,1}-V_{6,1}=-9$	$CT_{6,2}=U_{6,2}-V_{6,2}=-84$
$CT_{6,3}=U_{6,3}-V_{6,3}=3$	$CT_{6,4}=U_{6,4}-V_{6,4}=-14$	$CT_{6,5}=U_{6,5}-V_{6,5}=-61$
$CT_{7,1}=U_{7,1}-V_{7,1}=3$	$CT_{7,2}=U_{7,2}-V_{7,2}=-56$	$CT_{7,3}=U_{7,3}-V_{7,3}=-62$
$CT_{7,4}=U_{7,4}-V_{7,4}=-9$	$CT_{7,5}=U_{7,5}-V_{7,5}=-9$	$CT_{8,1}=U_{8,1}-V_{8,1}=3$
$CT_{8,2}=U_{8,2}-V_{8,2}=-24$	$CT_{8,3}=U_{8,3}-V_{8,3}=-100$	$CT_{8,4}=U_{8,4}-V_{8,4}=-116$
$CT_{8,5}=U_{8,5}-V_{8,5}=-106$	$CT_{9,1}=U_{9,1}-V_{9,1}=3$	$CT_{9,2}=U_{9,2}-V_{9,2}=-6$
$CT_{9,3}=U_{9,3}-V_{9,3}=-80$	$CT_{9,4}=U_{9,4}-V_{9,4}=-95$	$CT_{9,5}=U_{9,5}-V_{9,5}=-73$

The negative value indicates that the idle time is sufficient for the respective stage to be used as the required setup time while positive value indicates that additional time is required over the idle time to fulfill the setup time requirement. Finally, the makespan is calculated using equation (4.4b) as follows:

$$Makespan = \sum_{j=1}^{5} M_{1,j} + \sum_{i=2}^{10} M_{i,5} + \sum_{i=1}^{9} H_{i,5} + \sum_{j=0}^{5} T_{1,j} + \sum_{i=2}^{10} (T_{i,m-1} + T_{i,m}) = 1891 \text{ hours} \qquad (4.4b)$$

$$Where \sum_{i=1}^{9} H_{i,5} = \sum_{i=1}^{9} V_{i,5} + \sum_{i=1}^{9} MAX(CT_{i,1}, CT_{i,2}, CT_{i,3}, CT_{i,4}, CT_{i,5})$$

The makespan value obtained above can be conferred by representing the batch process recipe on a Gantt chart with arrangement shown below in agreement with the ZW transfer policy.

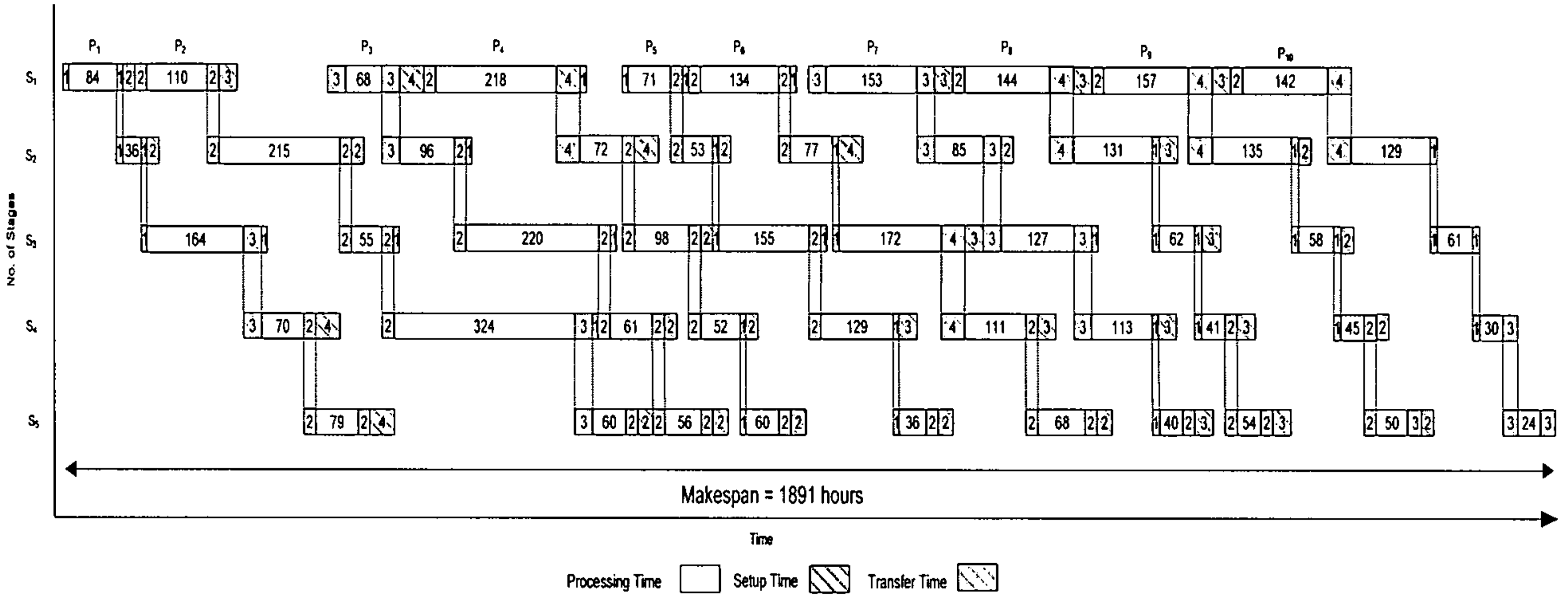


Figure: Gantt chart for production sequence $P_1P_2P_3P_4P_5P_6P_7P_8P_9P_{10}$ (ZW transfer policy)

NIS/UW transfer policy

The batch process recipe is represented to form a matrix $M_{i,j}$ to illustrate the arrangement of variables, V and I, that would determine the idle time and holding time respectively.

	j				
i	1	2	3	4	5
1	84	36	164	70	79
	$V_{1,1}$	$V_{1,2}$	$V_{1,3}$	$V_{1,4}$	$V_{1,5}$
2	110 $I_{1,1}$	215 $I_{1,2}$	55 $I_{1,3}$	324 $I_{1,4}$	60
	$V_{2,1}$	$V_{2,2}$	$V_{2,3}$	$V_{2,4}$	$V_{2,5}$
3	68 $I_{2,1}$	96 $I_{2,2}$	220 $I_{2,3}$	61 $I_{2,4}$	56
	$V_{3,1}$	$V_{3,2}$	$V_{3,3}$	$V_{3,4}$	$V_{3,5}$
4	218 $I_{3,1}$	72 $I_{3,2}$	98 $I_{3,3}$	52 $I_{3,4}$	60
	$V_{4,1}$	$V_{4,2}$	$V_{4,3}$	$V_{4,4}$	$V_{4,5}$
5	71 $I_{4,1}$	53 $I_{4,2}$	155 $I_{4,3}$	129 $I_{4,4}$	36
	$V_{5,1}$	$V_{5,2}$	$V_{5,3}$	$V_{5,4}$	$V_{5,5}$
6	134 $I_{5,1}$	77 $I_{5,2}$	172 $I_{5,3}$	111 $I_{5,4}$	68
	$V_{6,1}$	$V_{6,2}$	$V_{6,3}$	$V_{6,4}$	$V_{6,5}$
7	153 $I_{6,1}$	85 $I_{6,2}$	127 $I_{6,3}$	113 $I_{6,4}$	40
	$V_{7,1}$	$V_{7,2}$	$V_{7,3}$	$V_{7,4}$	$V_{7,5}$
8	144 $I_{7,1}$	131 $I_{7,2}$	62 $I_{7,3}$	41 $I_{7,4}$	54
	$V_{8,1}$	$V_{8,2}$	$V_{8,3}$	$V_{8,4}$	$V_{8,5}$
9	157 $I_{8,1}$	135 $I_{8,2}$	58 $I_{8,3}$	45 $I_{8,4}$	50
	$V_{9,1}$	$V_{9,2}$	$V_{9,3}$	$V_{9,4}$	$V_{9,5}$
10	142	129	61	30	24

Following are the equation developed to determine the inter-stage idle times with sequence dependent transfer times.

$$V_{1,j+1} = (V_{1,j} + M_{2,j} + T_{2,j-1}) - (M_{1,j+1} + T_{1,j+1}) \qquad j = 1........m-1 \qquad (4.5b)$$

$$V_{i,j+1} = (V_{i,j} + M_{i+1,j} + T_{i+1,j-1}) - (M_{i,j+1} + T_{i,j+1} + I_{i-1,j+1}) \quad j = 1........m-1, \quad i = 2.......n-1 \quad (4.6b)$$

$$V_{i,1} = 0, \quad I_{i,m} = 0 \qquad i = 1.......n-1$$

Next, a comparison is made between the variables, V, and the required setup times, U. The highest value from the comparison made is selected and represented using another variable, H.

$V_{1,1} = 0$	$U_{1,1} = 2$	$H_{1,1} = 2$
$V_{2,1} = 0$	$U_{2,1} = 3$	$H_{2,1} = 3$
$V_{3,1} = 0$	$U_{3,1} = 4$	$H_{3,1} = 4$
$V_{4,1} = 0$	$U_{4,1} = 1$	$H_{4,1} = 1$
$V_{5,1} = 0$	$U_{5,1} = 1$	$H_{5,1} = 1$
$V_{6,1} = 0$	$U_{6,1} = 1$	$H_{6,1} = 1$
$V_{7,1} = 0$	$U_{7,1} = 3$	$H_{7,1} = 3$
$V_{8,1} = 0$	$U_{8,1} = 3$	$H_{8,1} = 3$
$V_{9,1} = 0$	$U_{9,1} = 3$	$H_{9,1} = 3$

The variable $V_{1,2}$ for second stage between the first two products is determined using equation (4.5b). However, the variable $V_{1,1}$ in equation (4.5b) will be replaced with variable $H_{1,1}$ due to the comparison made above. Next, the calculated value for variable $V_{1,2}$ is compared with the required setup time $U_{1,2}$. The highest from the comparison made is selected and assigned as variable $H_{1,2}$ as shown below.

$V_{1,2} = (H_{1,1}+M_{2,1}+T_{2,0}) - (M_{1,2}+T_{1,2}) = 77,$ $U_{1,2}=2,$ $H_{1,2}= 77$

The calculation of holding time, $I_{1,1}$, in the earlier stage could be done using the value of $H_{1,2}$ calculated above using the following equation.

$$I_{1,j} = (H_{1,j+1} + M_{1,j+1} + T_{1,j+1}) - (H_{1,j} + M_{2,j} + T_{2,j-1}) \qquad j = 1 \ldots\ldots\ldots\ldots m-1 \qquad (5.1)$$

$I_{1,1} = (H_{1,2} + M_{1,2} + T_{1,2}) - (H_{1,1} + M_{2,1} + T_{2,0}) = 1.3$

Similarly $V_{1,3}$.........$V_{1,5}$ are calculated with respective $I_{1,2}$.......$I_{1,4}$ as follows:

$V_{1,3} = (H_{1,2} + M_{2,2} + T_{2,1}) - (M_{1,3} + T_{1,3}) = 127,$ $U_{1,3}=1,$ $H_{1,3}= 127$

$I_{1,2} = (H_{1,3} + M_{1,3} + T_{1,3}) - (H_{1,2} + M_{2,2} + T_{2,1}) = 0$

$V_{1,4}= (H_{1,3} + M_{2,3} + T_{2,2}) - (M_{1,4} + T_{1,4}) = 112,$ $U_{1,4}=4,$ $H_{1,4}= 112$

$I_{1,3} = (H_{1,4} + M_{1,4} + T_{1,4}) - (H_{1,3} + M_{2,3} + T_{2,2}) = 0$

$V_{1,5} = (H_{1,4} + M_{2,4} + T_{2,3}) - (M_{1,5} + T_{1,5}) = 357,$ $U_{1,5}=4,$ $H_{1,5}= 357$

$I_{1,4} = (H_{1,5} + M_{1,5} + T_{1,5}) - (H_{1,4} + M_{2,4} + T_{2,3}) = 0$

The variables between stages of the succeeding products are determined using equation (4.6b). Similar to the comparison procedure adopted earlier, the variable, V, is replaced with corresponding variable, H. The holding time, I, in the earlier stage is determined using equation (5.2) below.

$$I_{i,j} = (H_{i,j+1} + M_{i,j+1} + T_{i,j+1} + I_{i-1,j+1}) - (H_{i,j} + M_{i+1,j} + T_{i+1,j-1}) \quad j = 1.......m-1, \quad i = 2.......n-1 \tag{5.2}$$

$$I_{i,m} = 0 \qquad i = 1.........n-1$$

$V_{2,2} = (H_{2,1} + M_{3,1} + T_{3,0}) - (M_{2,2} + T_{2,2} + I_{1,2}) = 0$ $U_{2,2}=2,$ $H_{2,2}=2$

$I_{2,1} = (H_{2,2} + M_{2,2} + T_{2,2} + I_{1,2}) - (H_{2,1} + M_{3,1} + T_{3,0}) = 145$

$V_{2,3} = (H_{2,2} + M_{3,2} + T_{3,1}) - (M_{2,3} + T_{2,3} + I_{1,3})= 44$ $U_{2,3}=1,$ $H_{2,3}=44$

$I_{2,2} = (H_{2,3} + M_{2,3} + T_{2,3} + I_{1,3}) - (H_{2,2} + M_{3,2} + T_{3,1}) = 0$

$V_{2,4} = (H_{2,3} + M_{3,3}+T_{3,2}) - (M_{2,4} +T_{2,4}+I_{1,4})= 0$ $U_{2,4}=1,\ H_{2,4}=1$

$I_{2,3} = (H_{2,4} + M_{2,4} +T_{2,4}+ I_{1,4}) - (H_{2,3} + M_{3,3}+T_{3,2}) = 62$

$V_{2,5}= (H_{2,4} + M_{3,4}+T_{3,3}) - (M_{2,5} +T_{2,5}+I_{1,5})= 2$ $U_{2,5}=2,\ H_{2,5}=2$

$I_{2,4} = (H_{2,5} + M_{2,5} +T_{2,5}+ I_{1,5}) - (H_{2,4} + M_{3,4}+T_{3,3}) = 0$

$V_{3,2} = (H_{3,1} + M_{4,1}+T_{4,0}) - (M_{3,2} +T_{3,2}+ I_{2,2}) = 126$ $U_{3,2}=1,\ H_{3,2}=126$

$I_{3,1} = (H_{3,2} + M_{3,2} +T_{3,2}+ I_{2,2}) - (H_{3,1} + M_{4,1}+T_{4,0}) = 0$

$V_{3,3} = (H_{3,2} + M_{4,2}+T_{4,1}) - (M_{3,3} +T_{3,3}+I_{2,3}) = 0$ $U_{3,3}=1,\ H_{3,3}= 1$

$I_{3,2} = (H_{3,3} + M_{3,3} +T_{3,3}+ I_{2,3}) - (H_{3,2} + M_{4,2}+T_{4,1}) = 83$

$V_{3,4} = (H_{3,3} + M_{4,3}+T_{4,2}) - (M_{3,4} +T_{3,4}+I_{2,4}) = 38$ $U_{3,4}=2,\ H_{3,4}= 38$

$I_{3,3} = (H_{3,4} + M_{3,4} +T_{3,4}+ I_{2,4}) - (H_{3,3} + M_{4,3}+T_{4,2}) = 0$

$V_{3,5} = (H_{3,4} + M_{4,4}+T_{4,3}) - (M_{3,5} +T_{3,5}+I_{2,5}) = 34$ $U_{3,5}=2,\ H_{3,5}= 34$

$I_{3,4} = (H_{3,5} + M_{3,5} +T_{3,5}+ I_{2,5}) - (H_{3,4} + M_{4,4}+T_{4,3}) = 0$

$V_{4,2} = (H_{4,1} + M_{5,1}+T_{5,0}) - (M_{4,2} +T_{4,2}+I_{3,2}) = 0$ $U_{4,2}=4,\ H_{4,2}= 4$

$I_{4,1} = (H_{4,2} + M_{4,2} +T_{4,2}+ I_{3,2}) - (H_{4,1} + M_{5,1}+T_{5,0}) = 88$

$V_{4,3} = (H_{4,2} + M_{5,2}+T_{5,1}) - (M_{4,3} +T_{4,3}+I_{3,3}) = 0$ $U_{4,3}=2,\ H_{4,3}= 2$

$I_{4,2} = (H_{4,3} + M_{4,3} +T_{4,3}+ I_{3,3}) - (H_{4,2} + M_{5,2}+T_{5,1}) = 43$

$V_{4,4} = (H_{4,3} + M_{5,3}+T_{5,2}) - (M_{4,4} +T_{4,4}+I_{3,4}) = 105$ $U_{4,4}=2,\ H_{4,4}= 105$

$I_{4,3} = (H_{4,4} + M_{4,4} +T_{4,4}+ I_{3,4}) - (H_{4,3} + M_{5,3}+T_{5,2}) = 0$

$V_{4,5} = (H_{4,4} + M_{5,4}+T_{5,3}) - (M_{4,5} +T_{4,5}+I_{3,5}) = 174$ $U_{4,5}=2,\ H_{4,5}= 174$

$I_{4,4} = (H_{4,5} + M_{4,5} +T_{4,5}+ I_{3,5}) - (H_{4,4} + M_{5,4}+T_{5,3}) = 0$

$V_{5,2} = (H_{5,1} + M_{6,1}+T_{6,0}) - (M_{5,2} +T_{5,2}+I_{4,2}) = 40$ $U_{5,2}=2,\ H_{5,2}= 40$

$I_{5,1} = (H_{5,2} + M_{5,2} +T_{5,2}+ I_{4,2}) - (H_{5,1} + M_{6,1}+T_{6,0}) = 0$

$V_{5,3} = (H_{5,2} + M_{6,2}+T_{6,1}) - (M_{5,3} +T_{5,3}+I_{4,3}) = 0$ $U_{5,3}=1,\ H_{5,3}= 1$

$I_{5,2} = (H_{5,3} + M_{5,3} +T_{5,3}+ I_{4,3}) - (H_{5,2} + M_{6,2}+T_{6,1}) = 39$

$V_{5,4} = (H_{5,3} + M_{6,3}+T_{6,2}) - (M_{5,4} +T_{5,4}+I_{4,4}) = 44$ $U_{5,4}=3,\ H_{5,4}= 44$

$I_{5,3} = (H_{5,4} + M_{5,4} +T_{5,4}+ I_{4,4}) - (H_{5,3} + M_{6,3}+T_{6,2}) = 0$

$V_{5,5} = (H_{5,4} + M_{6,4}+T_{6,3}) - (M_{5,5} +T_{5,5}+I_{4,5}) = 121$ $U_{5,5}=2,\ H_{5,5}= 121$

$I_{5,4} = (H_{5,5} + M_{5,5} +T_{5,5}+ I_{4,5}) - (H_{5,4} + M_{6,4}+T_{6,3}) = 0$

$V_{6,2} = (H_{6,1} + M_{7,1}+T_{7,0}) - (M_{6,2} +T_{6,2}+I_{5,2}) = 40$ $U_{6,2}=4,\ H_{6,2}= 40$

$I_{6,1} = (H_{6,2} + M_{6,2} +T_{6,2}+ I_{5,2}) - (H_{6,1} + M_{7,1}+T_{7,0}) = 0$

$V_{6,3} = (H_{6,2} + M_{7,2}+T_{7,1}) - (M_{6,3} +T_{6,3}+I_{5,3}) = 0$ $U_{6,3}=3,\ H_{6,3}= 3$

$I_{6,2} = (H_{6,3} + M_{6,3} +T_{6,3}+ I_{5,3}) - (H_{6,2} + M_{7,2}+T_{7,1}) = 51$

$V_{6,4} = (H_{6,3} + M_{7,3}+T_{7,2}) - (M_{6,4} +T_{6,4}+I_{5,4}) = 20$ $U_{6,4}=3,\ H_{6,4}= 20$

$I_{6,3} = (H_{6,4} + M_{6,4} +T_{6,4}+ I_{5,4}) - (H_{6,3} + M_{7,3}+T_{7,2}) = 0$

$V_{6,5} = (H_{6,4} + M_{7,4}+T_{7,3}) - (M_{6,5} +T_{6,5}+I_{5,5}) = 66$ $U_{6,5}=2,\ H_{6,5}= 66$

$I_{6,4} = (H_{6,5} + M_{6,5} +T_{6,5}+ I_{5,5}) - (H_{6,4} + M_{7,4}+T_{7,3}) = 0$

$V_{7,2} = (H_{7,1} + M_{8,1}+T_{8,0}) - (M_{7,2} +T_{7,2}+I_{6,2}) = 10$ $U_{7,2}=2,\ H_{7,2}= 10$

$I_{7,1} = (H_{7,2} + M_{7,2} +T_{7,2}+ I_{6,2}) - (H_{7,1} + M_{8,1}+T_{8,0}) = 0$

$V_{7,3} = (H_{7,2} + M_{8,2}+T_{8,1}) - (M_{7,3} +T_{7,3}+I_{6,3}) = 15$ $U_{7,3}=1,\ H_{7,3}= 15$

$I_{7,2} = (H_{7,3} + M_{7,3} +T_{7,3}+ I_{6,3}) -(H_{7,2} + M_{8,2}+T_{8,1})= 0$

$V_{7,4} = (H_{7,3} + M_{8,3}+T_{8,2}) - (M_{7,4} +T_{7,4}+I_{6,4}) = 0$ $U_{7,4}=3,\ H_{7,4}= 3$

$I_{7,3} = (H_{7,4} + M_{7,4} +T_{7,4}+ I_{6,4}) - (H_{7,3} + M_{8,3}+T_{8,2}) = 39$

$V_{7,5} = (H_{7,4} + M_{8,4}+T_{8,3}) - (M_{7,5} +T_{7,5}+I_{6,5}) =0$ $U_{7,5}=3,\ H_{7,5}= 3$

$I_{7,4} = (H_{7,5} + M_{7,5} +T_{7,5}+ I_{6,5}) - (H_{7,4} + M_{8,4}+T_{8,3}) = 0$

$V_{8,2} = (H_{8,1} + M_{9,1}+T_{9,0}) - (M_{8,2} +T_{8,2}+I_{7,2}) = 30$ $U_{8,2}=3,\ H_{8,2}= 30$

$I_{8,1} = (H_{8,2} + M_{8,2} +T_{8,2}+ I_{7,2}) - (H_{8,1} + M_{9,1}+T_{9,0}) = 0$

$V_{8,3} = (H_{8,2} + M_{9,2}+T_{9,1}) - (M_{8,3} +T_{8,3}+I_{7,3}) = 67$ $U_{8,3}=3,\ H_{8,3}= 67$

$I_{8,2} = (H_{8,3} + M_{8,3} +T_{8,3}+ I_{7,3}) - (H_{8,2} + M_{9,2}+T_{9,1}) = 0$

$V_{8,4} = (H_{8,3} + M_{9,3}+T_{9,2}) - (M_{8,4} +T_{8,4}+I_{7,4}) =83$ $U_{8,4}=3,\ H_{8,4}= 83$

$I_{8,3} = (H_{8,4} + M_{8,4} +T_{8,4}+ I_{7,4}) - (H_{8,3} + M_{9,3}+T_{9,2}) = 0$

$V_{8,5} = (H_{8,4} + M_{9,4}+T_{9,3}) - (M_{8,5} +T_{8,5}+I_{7,5}) = 73$ $U_{8,5}=3,\ H_{8,5}= 73$

$I_{8,4} = (H_{8,5} + M_{8,5} +T_{8,5}+ I_{7,5}) - (H_{8,4} + M_{9,4}+T_{9,3}) = 0$

$V_{9,2} = (H_{9,1} + M_{10,1}+T_{10,0}) - (M_{9,2} +T_{9,2}+I_{8,2}) = 11$ $U_{9,2}=2,\ H_{9,2}= 11$

$I_{9,1} = (H_{9,2} + M_{9,2} +T_{9,2}+ I_{8,2}) - (H_{9,1} + M_{10,1}+T_{10,0})= 0$

$V_{9,3} = (H_{9,2} + M_{10,2}+T_{10,1}) - (M_{9,3} +T_{9,3}+I_{8,3}) = 85$ $U_{9,3}=2,\ H_{9,3}= 85$

$I_{9,2} = (H_{9,3} + M_{9,3} +T_{9,3}+ I_{8,3}) - (H_{9,2} + M_{10,2}+T_{10,1}) = 0$

$V_{9,4} = (H_{9,3} + M_{10,3}+T_{10,2}) - (M_{9,4} +T_{9,4}+I_{8,4}) = 100$ $U_{9,4}=2,\ H_{9,4}= 100$

$I_{9,3} = (H_{9,4} + M_{9,4} +T_{9,4}+ I_{8,4}) - (H_{9,3} + M_{10,3}+T_{10,2}) = 0$

$V_{9,5} = (H_{9,4} + M_{10,4}+T_{10,3}) - (M_{9,5} +T_{9,5}+I_{8,5}) = 78$ $U_{9,5}=2,\ H_{9,5}= 78$

$I_{9,4} = (H_{9,5} + M_{9,5} +T_{9,5}+ I_{8,5}) - (H_{9,4} + M_{10,4}+T_{10,3}) = 0$

Finally, the makespan is calculated using equation (4.4c) as follows:

$$Makespan = \sum_{j=1}^{m} M_{1,j} + \sum_{i=2}^{n} M_{i,m} + \sum_{i=1}^{n-1} H_{i,m} + \sum_{j=0}^{m} T_{1,j} + \sum_{i=2}^{n} (T_{i,m-1} + T_{i,m}) = 1836 \text{ hours} \qquad (4.4c)$$

Similar to the ZW transfer policy, a Gantt chart is shown below to confer the value of the makespan obtained above. The arrangement of the batch process recipe on the Gantt chart is in accordance with NIS/UW transfer policy.

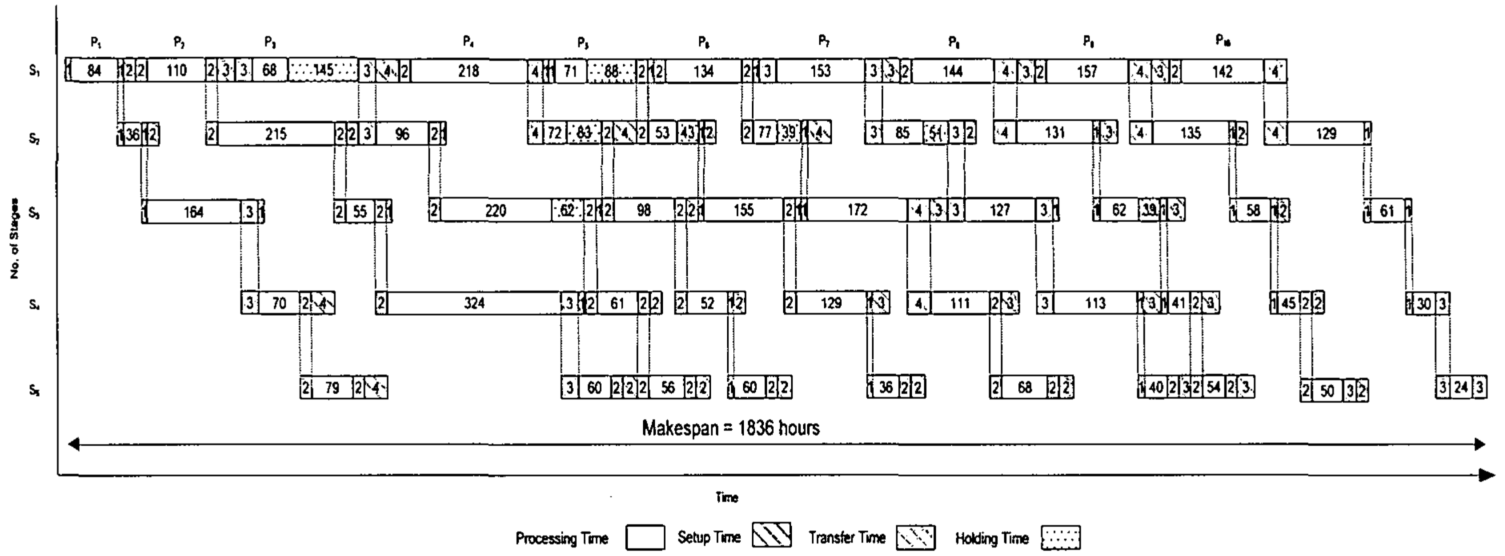


Figure: Gantt chart for production sequence $P_1P_2P_3P_4P_5P_6P_7P_8P_9P_{10}$(NIS/UW transfer policy)

APPENDIX B

Summary of mathematical formulations developed for makespan determination without transfer and setup time for batch processes operated under various transfer policies

1. ZW Transfer Policy

$$V_{i,2} = M_{i+1,1} - M_{i,2} \qquad i = 1......n\text{-}1 \qquad (4.1)$$

$$V_{i,j+1} = (V_{i,j} + M_{i+1,j}) - M_{i,j+1} \qquad j = 2........m\text{-}1,\ \ i = 1......n\text{-}1 \qquad (4.2)$$

$$V_{i,j} = (V_{i,j+1} - M_{i+1,j}) + M_{i,j+1} \qquad j = m\text{-}1........1,\ \ i = 1......n\text{-}1 \qquad (4.3)$$

$$Makespan = \sum_{j=1}^{m} M_{1,j} + \sum_{i=2}^{n} M_{i,m} + \sum_{i=1}^{n-1} V_{i,m} \qquad n \geq 2\ ,\ m \geq 2 \qquad (4.4)$$

2. NIS/UW Transfer Policy

$$V_{i,1} = 0,\ \ I_{i,m} = 0 \qquad i = 1.......n\text{-}1$$

$$V_{1,j+1} = (V_{1,j} + M_{2,j}) - M_{1,j+1} \qquad j = 1........m-1 \qquad (4.5)$$

$$V_{i,j+1} = (V_{i,j} + M_{i+1,j}) - (M_{i,j+1} + I_{i-1,j+1}) \qquad j = 1........m-1,\ \ i = 2.......n\text{-}1 \qquad (4.6)$$

$$Makespan = \sum_{j=1}^{m} M_{1,j} + \sum_{i=2}^{n} M_{i,m} + \sum_{i=1}^{n-1} V_{i,m} \qquad n \geq 2\ ,\ m \geq 2 \qquad (4.4)$$

3. UIS/UW Transfer Policy

$$V_{i,1} = 0 \qquad i = 1......n-1$$

$$V_{1,j} = (M_{2,j-1} + V_{1,j-1}) - M_{1,j} \qquad j = 2......m \qquad (4.7)$$

$$V_{i,j} = (\sum_{k=2}^{i+1} M_{k,j-1} + \sum_{k=1}^{i} V_{k,j-1}) - (\sum_{k=1}^{i} M_{k,j} + \sum_{k=1}^{i-1} V_{k,j}) \quad j = 2.......m, \quad i = 2........n-1 \qquad (4.8)$$

$$Makespan = \sum_{j=1}^{m} M_{1,j} + \sum_{i=2}^{n} M_{i,m} + \sum_{i=1}^{n-1} V_{i,m} \qquad n \geq 2 \ , \ m \geq 2 \qquad (4.4)$$

4. FIS/UW Transfer Policy

$$V_{i,1} = 0, \ \ I_{1,j} = 0 \qquad i = 1..........n\text{-}1, \ \ j = 1.........m$$

$$I_{i,m} = 0 \qquad i = 2..........n\text{-}1$$

$$V_{1,j} = (M_{2,j-1} + V_{1,j-1}) - M_{1,j} \qquad j = 2........m \qquad (4.9)$$

$$\left[\begin{array}{l} if \ \ (M_{i+1,j-1} + V_{i,j-1}) \leq W_{i-1,j-1}, \ I_{i,j-1} = W_{i-1,j-1} - (M_{i+1,j-1} + V_{i,j-1}) \ \ else \ I_{i,j-1} = 0 \\ \\ V_{i,j} = (\sum_{k=2}^{i+1} M_{k,j-1} + \sum_{k=1}^{i} V_{k,j-1} + \sum_{k=1}^{i} I_{k,j-1}) - (\sum_{k=1}^{i} M_{k,j} + \sum_{k=1}^{i-1} V_{k,j} + \sum_{k=1}^{i-1} I_{k,j}) \\ \\ \qquad\qquad\qquad\qquad\qquad j = 2......m, \ i = 2......n\text{-}1 \end{array}\right] \qquad (4.10)$$

$$Makespan = \sum_{j=1}^{m} M_{1,j} + \sum_{i=2}^{n} M_{i,m} + \sum_{i=1}^{n-1} V_{i,m} \qquad n \geq 2 \ , \ m \geq 2 \qquad (4.4)$$

5. MIS/UW Transfer Policy

$$V_{i,1} = 0, \ \ I_{i,m} = 0 \qquad i = 1.......n\text{-}1$$

$$V_{1,j+1} = (V_{1,j} + M_{2,j}) - M_{1,j+1} \qquad j = 1........m-2 \qquad (4.5a)$$

$$V_{i,j+1} = (V_{i,j} + M_{i+1,j}) - (M_{i,j+1} + I_{i-1,j+1}) \qquad j = 1........m-2, \ \ i = 2.......n\text{-}1 \qquad (4.6a)$$

$$V_{1,j} = (M_{2,j-1} + V_{1,j-1}) - M_{1,j} \qquad j = m \qquad (4.7a)$$

$$V_{i,j} = (\sum_{k=2}^{i+1} M_{k,j-1} + \sum_{k=1}^{i} V_{k,j-1}) - (\sum_{k=1}^{i} M_{k,j} + \sum_{k=1}^{i-1} V_{k,j}) \quad j = m, \qquad i = 2........n-1 \qquad (4.8a)$$

$$Makespan = \sum_{j=1}^{m} M_{1,j} + \sum_{i=2}^{n} M_{i,m} + \sum_{i=1}^{n-1} V_{i,m} \qquad n \geq 2 \ , m \geq 2 \qquad (4.4)$$

APPENDIX C

Summary of mathematical formulations developed for makespan determination with transfer and setup time for batch processes operated under various transfer policies

1. ZW Transfer Policy

$$V_{i,2} = (M_{i+1,1} + T_{i+1,0}) - (M_{i,2} + T_{i,2}) \qquad i = 1......n\text{-}1. \qquad (4.1a)$$

$$V_{i,j+1} = (V_{i,j} + M_{i+1,j} + T_{i+1,j-1}) - (M_{i,j+1} + T_{i,j+1}) \qquad j = 2........m\text{-}1,\ \ i = 1......n\text{-}1 \qquad (4.2a)$$

$$V_{i,j} = (V_{i,j+1} - (M_{i+1,j} + T_{i+1,j-1})) + (M_{i,j+1} + T_{i,j+1}) \qquad j = m\text{-}1........1,\ \ i = 1......n\text{-}1 \qquad (4.3a)$$

$$Makespan = \sum_{j=1}^{m} M_{1,j} + \sum_{i=2}^{n} M_{i,m} + \sum_{i=1}^{n-1} H_{i,m} + \sum_{j=0}^{m} T_{1,j} + \sum_{i=2}^{n} (T_{i,m-1} + T_{i,m}) \qquad (4.4b)$$

$$Where \sum_{i=1}^{n-1} H_{i,m} = \sum_{i=1}^{n-1} V_{i,m} + \sum_{i=1}^{n-1} MAX(CT_{i,1}, CT_{i,2}, ...CT_{i,m})$$

2. NIS/UW Transfer Policy

$$V_{1,j+1} = (V_{1,j} + M_{2,j} + T_{2,j-1}) - (M_{1,j+1} + T_{1,j+1}) \qquad j = 1........m-1 \qquad (4.5a)$$

$$V_{i,j+1} = (V_{i,j} + M_{i+1,j} + T_{i+1,j-1}) - (M_{i,j+1} + T_{i,j+1} + I_{i-1,j+1}) \ \ j = 1........m-1,\ \ i = 2.......n\text{-}1 \ \ (4.6a)$$

$$V_{i,1} = 0,\ \ I_{i,m} = 0 \qquad i = 1.......n\text{-}1$$

$$Makespan = \sum_{j=1}^{m} M_{1,j} + \sum_{i=2}^{n} M_{i,m} + \sum_{i=1}^{n-1} H_{i,m} + \sum_{j=0}^{m} T_{1,j} + \sum_{i=2}^{n} (T_{i,m-1} + T_{i,m}) \qquad (4.4c)$$

3. UIS/UW Transfer Policy

$$V_{i,1} = 0 \qquad i = 1......n-1$$

$$V_{1,j} = (M_{2,j-1} + V_{1,j-1}) - M_{1,j} \qquad j = 2......m \tag{4.7}$$

$$\begin{bmatrix} \mathit{if}\ H_{1,j+1} > V_{1,j+1},\ \ W_{1,j} = (H_{1,j+1} + M_{1,j+1}) - (H_{1,j} + M_{2,j} + T_{2,j})\ \ \mathit{else}\ \ W_{1,j} = 0 \\ \mathit{if}\ (H_{1,j+1} > V_{1,j+1}\ \ \mathit{and}\ \ W_{1,j} \le 0),\ \ \ H_{1,j+1} = H_{1,j+1} + (-W_{1,j})\ \ \mathit{and}\ \ W_{1,j} = 0\ (\mathit{pass}) \end{bmatrix}$$

$$j = 1.........m\text{-}1 \tag{5.3}$$

$$V_{i,j} = (\sum_{k=2}^{i+1} M_{k,j-1} + \sum_{k=2}^{i} T_{k,j-1} + \sum_{k=1}^{i} H_{k,j-1}) - (\sum_{k=1}^{i} M_{k,j} + \sum_{k=2}^{i} T_{k,j} + \sum_{k=1}^{i-1} H_{k,j})$$

$$j = 2.......m,\quad i = 2........n-1 \tag{4.8b}$$

$$\begin{bmatrix} \mathit{if}\ H_{i,j} > V_{i,j},\ W_{i,j-1} = (\sum_{k=1}^{i} M_{k,j} + \sum_{k=2}^{i} T_{k,j} + \sum_{k=1}^{i} H_{k,j}) - (\sum_{k=2}^{i+1} M_{k,j-1} + \sum_{k=2}^{i+1} T_{k,j-1} + \sum_{k=1}^{i} H_{k,j-1}) \\ \mathit{else}\ W_{i,j-1}{=}0 \\ \mathit{if}\ (H_{i,j} > V_{i,j}\ \ \mathit{and}\ \ W_{i,j-1} \le 0),\ \ \ H_{i,j} = H_{i,j} + (-W_{i,j-1})\ \ \mathit{and}\ \ W_{i,j-1} = 0\ (\mathit{pass}) \end{bmatrix}$$

$$j = 2.......m,\quad i = 2........n-1 \tag{5.4}$$

$$\mathit{Makespan} = \sum_{j=1}^{m} M_{1,j} + \sum_{i=2}^{n} M_{i,m} + \sum_{i=1}^{n-1} H_{i,m} + \sum_{i=2}^{n} T_{i,m} \tag{4.4d}$$

4. FIS/UW Transfer Policy

$$V_{i,1} = 0,\ \ I_{1,j} = 0 \qquad i = 1.........n\text{-}1,\ \ j = 1.........m$$

$$I_{i,m} = 0 \qquad i = 2.........n\text{-}1$$

$$V_{1,j} = (M_{2,j-1} + V_{1,j-1}) - M_{1,j} \qquad j = 2........m \qquad (4.9)$$

$$\left[\begin{array}{l} \text{if } H_{1,j+1} > V_{1,j+1},\ \ W_{1,j} = (H_{1,j+1} + M_{1,j+1}) - (H_{1,j} + M_{2,j} + T_{2,j}) \ \text{ else } \ W_{1,j} = 0 \\ \\ \text{if } (H_{1,j+1} > V_{1,j+1} \ \text{ and } \ W_{1,j} \le 0),\quad H_{1,j+1} = H_{1,j+1} + (-W_{1,j}) \ \text{ and } \ W_{1,j} = 0 \ (pass) \end{array}\right]$$

$$j = 1.........m\text{-}1 \qquad (5.5)$$

$$\left[\begin{array}{l} \text{if } G_{i,j-1} \le (M_{i+1,j-1} + H_{i,j-1}) - (W_{i-1,j-1} + T_{i,j-1}),\ G_{i,j-1} = 0 \\ \\ \text{if } \ (M_{i+1,j-1} + H_{i,j-1}) \le (W_{i-1,j-1} + T_{i,j-1} + G_{i,j-1}),\ I_{i,j-1} = (W_{i-1,j-1} + T_{i,j-1} + G_{i,j-1}) - (M_{i+1,j-1} + H_{i,j-1}) \\ \text{else } I_{i,j-1} = 0 \\ \\ V_{i,j} = (\sum_{k=2}^{i+1} M_{k,j-1} + \sum_{k=2}^{i} T_{k,j-1} + \sum_{k=1}^{i} H_{k,j-1} + \sum_{k=1}^{i} I_{k,j-1}) - (\sum_{k=1}^{i} M_{k,j} + \sum_{k=2}^{i} T_{k,j} + \sum_{k=1}^{i-1} H_{k,j} + \sum_{k=1}^{i-1} I_{k,j}) \end{array}\right]$$

$$j = 2......m,\ i = 2......n\text{-}1 \qquad (4.10a)$$

$$\left[\begin{array}{l} \text{if } H_{i,j} > V_{i,j},\ W_{i,j-1} = (\sum_{k=1}^{i} M_{k,j} + \sum_{k=2}^{i} T_{k,j} + \sum_{k=1}^{i} H_{k,j} + \sum_{k=1}^{i-1} I_{k,j}) - (\sum_{k=2}^{i+1} M_{k,j-1} + \sum_{k=2}^{i+1} T_{k,j-1} + \sum_{k=1}^{i} H_{k,j-1} + \sum_{k=1}^{i} I_{k,j-1}) \\ \text{else } W_{i,j-1} = 0 \\ \\ \text{if } (H_{i,j} > V_{i,j} \ \text{ and } \ W_{i,j-1} \le 0),\quad H_{i,j} = H_{i,j} + (-W_{i,j-1}) \ \text{ and } \ W_{i,j-1} = 0 \ (pass) \end{array}\right]$$

$$j = 2......m,\ i = 2......n\text{-}1 \qquad (5.6)$$

$$I_{1,j} = 0,\ \ I_{i,m} = 0 \qquad i = 1.........n\text{-}1,\ j = 1........m$$

$$Makespan = \sum_{j=1}^{m} M_{1,j} + \sum_{i=2}^{n} M_{i,m} + \sum_{i=1}^{n-1} H_{i,m} + \sum_{i=2}^{n} T_{i,m} \qquad (4.4d)$$

5. MIS/UW Transfer Policy

$$V_{i,1}=0,\ I_{i,m}=0 \qquad i=1.......n-1$$

$$V_{1,j+1}=(V_{1,j}+M_{2,j}+T_{2,j-1})-(M_{1,j+1}+T_{1,j+1}) \qquad j=1........m-2 \qquad (4.5c)$$

$$I_{1,j}=(H_{1,j+1}+M_{1,j+1}+T_{1,j+1})-(H_{1,j}+M_{2,j}+T_{2,j-1}) \qquad j=1............m-2 \qquad (5.1a)$$

$$V_{i,j+1}=(V_{i,j}+M_{i+1,j}+T_{i+1,j-1})-(M_{i,j+1}+T_{i,j+1}+I_{i-1,j+1}) \quad j=1........m-2,\ \ i=2.......n-1 \quad (4.6c)$$

$$I_{i,j}=(H_{i,j+1}+M_{i,j+1}+T_{i,j+1}+I_{i-1,j+1})-(H_{i,j}+M_{i+1,j}+T_{i+1,j-1})$$
$$j=1.......m-2,\quad i=2.......n-1 \qquad (5.2a)$$

$$V_{1,j}=(M_{2,j-1}+V_{1,j-1})-M_{1,j} \qquad j=m \qquad (4.7a)$$

$$\begin{bmatrix} \textit{if}\ H_{1,j+1}>V_{1,j+1},\ \ W_{1,j}=(H_{1,j+1}+M_{1,j+1})-(H_{1,j}+M_{2,j}+T_{2,j})\ \ \textit{else}\ \ W_{1,j}=0 \\ \\ \textit{if}\ (H_{1,j+1}>V_{1,j+1}\ \ \textit{and}\ \ W_{1,j}\le 0),\quad H_{1,j+1}=H_{1,j+1}+(-W_{1,j})\ \ \textit{and}\ \ W_{1,j}=0\ (\textit{pass}) \end{bmatrix}$$
$$j=m-1 \qquad (5.3a)$$

$$V_{i,j}=(\sum_{k=2}^{i+1}M_{k,j-1}+\sum_{k=2}^{i}T_{k,j-1}+\sum_{k=1}^{i}H_{k,j-1})-(\sum_{k=1}^{i}M_{k,j}+\sum_{k=2}^{i}T_{k,j}+\sum_{k=1}^{i-1}H_{k,j})$$
$$j=m,\quad i=2........n-1 \qquad (4.8c)$$

$$\begin{bmatrix} \textit{if}\ H_{i,j}>V_{i,j},\ W_{i,j-1}=(\sum_{k=1}^{i}M_{k,j}+\sum_{k=2}^{i}T_{k,j}+\sum_{k=1}^{i}H_{k,j})-(\sum_{k=2}^{i+1}M_{k,j-1}+\sum_{k=2}^{i+1}T_{k,j-1}+\sum_{k=1}^{i}H_{k,j-1}) \\ \\ \textit{else}\ W_{i,j-1}{=}0 \\ \\ \textit{if}\ (H_{i,j}>V_{i,j}\ \ \textit{and}\ \ W_{i,j-1}\le 0),\quad H_{i,j}=H_{i,j}+(-W_{i,j-1})\ \ \textit{and}\ \ W_{i,j-1}=0\ (\textit{pass}) \end{bmatrix}$$
$$j=m,\quad i=2........n-1 \qquad (5.4a)$$

$$\textit{Makespan}=\sum_{j=1}^{m}M_{1,j}+\sum_{i=2}^{n}M_{i,m}+\sum_{i=1}^{n-1}H_{i,m}+\sum_{i=2}^{n}T_{i,m} \qquad (4.4d)$$

APPENDIX D

n = 7, m = 5

Transfer Policy	CPU Time (sec)					
	1	2	3	4	5	Mean
ZW	1.723	1.734	1.743	1.723	1.736	1.7318
NIS	1.713	1.732	1.713	1.734	1.713	1.721
UIS	1.743	1.743	1.744	1.753	1.743	1.7452
FIS	1.743	1.745	1.743	1.745	1.743	1.7438
MIS	1.743	1.745	1.723	1.756	1.732	1.7398

n = 7, m = 7

Transfer Policy	CPU Time (sec)					
	1	2	3	4	5	Mean
ZW	1.953	1.953	1.952	1.963	1.953	1.9548
NIS	1.933	1.934	1.932	1.933	1.936	1.9336
UIS	1.985	1.976	1.989	1.988	1.989	1.9854
FIS	1.963	1.946	1.964	1.962	1.960	1.959
MIS	1.948	1.946	2.049	1.949	1.940	1.9664

n = 7, m = 9

Transfer Policy	CPU Time (sec)					
	1	2	3	4	5	Mean
ZW	2.113	2.113	2.116	2.112	2.113	2.1134
NIS	2.103	2.103	2.104	2.108	2.103	2.1042
UIS	2.544	2.650	2.897	2.699	2.824	2.7228
FIS	2.143	2.143	2.145	2.142	2.143	2.1432
MIS	2.183	2.193	2.193	2.188	2.183	2.188

n = 8, m = 5

Transfer Policy	CPU Time (sec)					
	1	2	3	4	5	Mean
ZW	20.554	20.558	20.554	20.556	20.550	20.5544
NIS	20.724	20.745	20.734	20.749	20.753	20.741
UIS	20.804	20.842	20.904	20.914	20.904	20.8736
FIS	20.844	20.845	20.842	20.846	20.848	20.845
MIS	20.432	20.875	20.475	20.624	20.632	20.6076

n = 8, m = 7

Transfer Policy	CPU Time (sec)					
	1	2	3	4	5	Mean
ZW	21.683	21.693	21.783	21.783	21.763	21.741
NIS	21.024	21.054	21.024	21.024	21.124	21.05
UIS	21.693	21.693	21.692	21.694	21.693	21.693
FIS	21.215	21.218	21.253	21.215	21.215	21.2232
MIS	21.528	21.545	21.669	21.656	21.724	21.6244

n = 8, m = 9

Transfer Policy	CPU Time (sec)					
	1	2	3	4	5	Mean
ZW	22.924	22.942	22.924	22.984	22.824	22.9196
NIS	22.794	22.790	22.794	22.694	22.594	22.7332
UIS	24.435	24.452	24.423	24.454	24.485	24.4498
FIS	22.645	22.652	22.658	22.645	22.652	22.6504
MIS	23.567	23.765	23.664	23.786	23.856	23.7276

n = 9, m = 5

Transfer Policy	CPU Time (sec)					
	1	2	3	4	5	Mean
ZW	207.861	207.861	207.864	207.872	207.861	207.864
NIS	206.592	206.594	206.597	206.292	206.592	206.533
UIS	207.148	207.148	207.182	207.843	207.482	207.361
FIS	204.284	204.294	204.238	204.285	204.248	204.27
MIS	207.206	207.226	207.268	207.222	207.206	207.226

n = 9, m = 7

Transfer Policy	CPU Time (sec)					
	1	2	3	4	5	Mean
ZW	217.432	217.245	217.342	217.546	217.441	217.401
NIS	213.955	213.985	213.950	213.955	213.952	213.959
UIS	216.314	216.314	216.349	217.348	217.378	216.741
FIS	215.331	215.442	215.745	215.586	215.745	215.57
MIS	216.432	216.245	216.134	216.246	216.446	216.301

n = 9, m = 9

Transfer Policy	CPU Time (sec)					
	1	2	3	4	5	Mean
ZW	219.308	219.358	219.348	220.386	219.358	219.552
NIS	218.16	218.28	218.20	219.62	218.16	218.484
UIS	221.142	221.232	221.345	220.671	220.991	221.076
FIS	220.206	220.266	220.218	220.206	220.345	220.248
MIS	220.101	221.123	220.345	220.171	221.191	220.586

n = 10, m = 5

Transfer Policy	CPU Time (sec)					
	1	2	3	4	5	Mean
ZW	2077.851	2077.882	2077.872	2077.859	2077.851	2077.86
NIS	2077.782	2077.782	2077.789	2077.782	2077.782	2077.78
UIS	2080.540	2080.540	2080.540	2080.540	2080.540	2080.54
FIS	2078.542	2078.542	2078.788	2078.878	2078.542	2078.66
MIS	2080.423	2080.546	2080.457	2080.422	2080.425	2080.45

n = 10, m = 7

Transfer Policy	CPU Time (sec)					
	1	2	3	4	5	Mean
ZW	2110.213	2110.235	2110.213	2110.286	2110.213	2110.23
NIS	2100.214	2100.888	2100.245	2100.214	2100.214	2100.36
UIS	2112.234	2112.234	2112.248	2112.247	2112.234	2112.24
FIS	2108.543	2108.547	2108.548	2108.586	2108.989	2108.64
MIS	2110.542	2110.572	2110.522	2110.512	2110.989	2110.63

n = 10, m = 9

Transfer Policy	CPU Time (sec)					
	1	2	3	4	5	Mean
ZW	2280.459	2280.659	2280.489	2280.444	2280.789	2280.57
NIS	2285.467	2285.476	2285.786	2285.123	2285.898	2285.55
UIS	2295.456	2295.468	2295.689	2295.569	2295.456	2295.53
FIS	2278.234	2278.788	2278.234	2278.234	2278.989	2278.5
MIS	2283.324	2283.823	2283.347	2283.257	2283.823	2283.51

APPENDIX E

```
// Program code developed in Microsoft Visual C++(R) 6.0 for the purpose
// of calculation of makespan for each production sequence resulted from
// the permutation of given product recipe. The results thus obtained
// are screened for the optimal solution to generate a list of produciton
// sequences with minimum makespan. The program code also displays
// inter-stage idle times, holding time inside stage and waiting time
// inside temporary storage for the respective transfer policy

    #include <iostream>           // allows use of input and output statements

    #include<vector>              // allows use of vectors

    #include <fstream>            // allows to create an output file

    #include<algorithm>           // allows use of permutation function

    #include <ctime>              // allows use of clock to determine CPU time

    using namespace std;          // contains C++ standard names

    #define ZW 1                  // assign number 1 to ZW

    #define NI 2                  // assign number 2 to NIS

    #define UI 3                  // assign number 3 to UIS

    #define FI 4                  // assign number 4 to FIS

    #define MI 5                  // assign number 5 to MIS

    #define  MAXX 100             // reserve memory location

    int MergeIndex1[100][2],      //array to merge two arrays

        MergeIndex2[100][2],      //array to merge two arrays

        MergeIndex[100],          //array to merge two arrays

        MergeK=0,                 //counter for merger

        counter3=1,               // counter for merged array

        Minrow1=0,                // number of minimum rows

        Mincol1=0,                 // number of minimum column sums

        rows = 1,                 // for rows sorting

        cols = 1,                 // for column sorting

        z1,                       // display merged array

        i=0,                      // number of rows

        j=0,                      // number of columns
        p=0,                    // number of poducts
        g=0,                    // number of stages
        k=0,                    // a counter
        kk=0,                   // a counter
        counter2=0,             // a counter
        policy=0,               // represents transfer policy
        method=0;               // selects enumeration method
        double temp;            // temporary variable
    float M[100][100],          // array for storing product recipe
          MM[100][100],         // array to restore position of initial data
          V[MAXX][MAXX],        // array for inter-stage idle times
          W[MAXX][MAXX],        // array for waiting times for temporary stor
age
          I[MAXX][MAXX],        // array for holding time inside stage
          CT[MAXX][MAXX],       // array for difference of idle time and setu
p time
          Mt[MAXX][MAXX],       // array for stage setup time
          WW[MAXX][MAXX],       // array for waiting times for temporary stor
age
          MN[MAXX][MAXX],       // array for production sequence
          MAKE=0.0,             // makespan variable
          min=0.0;              // variable for minimum value
    typedef vector <float> stages;  // vector definition
    vector <stages> ca;             // vector for processing time data
    vector <float> permt;           // vector to represent row number
    vector <stages> Q;              // vector for stage setup time
    vector <stages> t;              // vector for stage setup time
    ofstream myfile;                // output to an external file
    class Stopwatch {               // function to calculate CPU time
    public:
    Stopwatch( );
    void reset( );
    float elapsedTime( );

    private:

    clock_t initialObservedTime;

    };

        Stopwatch::Stopwatch ( ) {

        initialObservedTime = clock();

    }

    void Stopwatch::reset ( ) {

    initialObservedTime = clock();

    }

    float Stopwatch::elapsedTime ( ) {

        float time=0.0;

        return ((float) (clock( ) - initialObservedTime))/CLOCKS_PER_SEC;
    }

// functions for sorting rows and column

    int sortArray(int row)
        {
            int i, j, temp;

            for (i = (row); i >= 1; i--)
            {
               for (j = 1; j <= i; j++)
               {
                  if (MergeIndex1[j-1][0] > MergeIndex1[j][0])
                  {

                      temp = MergeIndex1[j-1][0];

                      MergeIndex1[j-1][0] = MergeIndex1[j][0];

                      MergeIndex1[j][0] = temp;

                      temp = MergeIndex1[j-1][1];

                      MergeIndex1[j-1][1] = MergeIndex1[j][1];

                      MergeIndex1[j][1] = temp;
                  }
               }
            }

            return 0;
    }

int sortArray2(int row)
        {
            int i, j, temp;
            for (i = (row); i >= 1; i--)
            {
               for (j = 1; j <= i; j++)
               {
                  if (MergeIndex2[j-1][0] > MergeIndex2[j][0])
                  {
                   temp = MergeIndex2[j-1][0];

                   MergeIndex2[j-1][0] = MergeIndex2[j][0];

                   MergeIndex2[j][0] = temp;

                   temp = MergeIndex2[j-1][1];

                   MergeIndex2[j-1][1] = MergeIndex2[j][1];

                   MergeIndex2[j][1] = temp;
                  }
               }
            }

            return 0;
    }

// function to calculate the number of possible production sequences from giv
en number of products

    double factor(int num) {

    if(num <= 1) return 1;

        temp = num * factor(num - 1);

        return temp;
    }

// function to convert processing time data from array to vector

    void ArrayToVector(){

    for(int i=1;i<=p;++i){

        vector<float> temp;

        for(int j =0;j<= ((g*3)+1);++j) temp.push_back(M[i][j]);

            ca.push_back(temp);

        }
    }

// function to convert row number of processing time data from array to vecto
r

    void ArrayToVectornew(){

        permt.clear();
        for(int i=1;i<=p;++i){
            permt.push_back(M[i][0]);
        }
    }
// function to convert processing time data from vector to array
    void VectorToArray(){
    int i=1;
    for(vector<stages>::iterator stit= ca.begin();stit!= ca.end();++stit)
        { int j=0;
             for(vector <float>::iterator it = stit->begin();it != stit->end(
);++it)
                M[i][j++] = *it;
                ++i;
        }
    }

// function to convert stage setup time data from vector to array
   void VectorToArraynew(vector <stages> ttemp,float Mtemp[MAXX][MAXX] ){
   int i=1;
   for(vector<stages>::iterator stit= ttemp.begin();stit!= ttemp.end();++stit
)
   { int j=0;
     for(vector <float>::iterator it = stit->begin();it != stit->end();++it)
        Mtemp[i][j++] = *it;
        ++i;
        }
    }
    float zmycode(),     // ZW algorithm
          nmycode(),     // NIS algorithm
          umycode(),     // UIS algorithm
          cmycode(),     // FIS algorithm
          mmycode(),     // MIS algorithm
          data(),        // function to display idle times for ZW/NIS/UIS/FIS
          ntank(),       // function for NIS data display
          mntank(),       // function for MIS data display

          utank(),        // function for UIS data display

          mutank(),       // function for MIS data display

          ctank(),        // function for FIS data display

          uiscis(),       // function to add processing time to transfer time i
n UIS/FIS

          uiscis2();      // function to exclude transfer time from processing
time in UIS/FIS


          int prog(float (*fp)());

    float (*fp)();


// function for permutation of stage setup time data in accordance with the p
ermutation of processing time data

    vector <stages> displaypermuQ(vector <stages>  q,vector<float> perm){

    vector <float>::iterator it1,it2;

    vector <stages> qTemp;

    vector <float> temp;

    for(vector <float>::iterator it_ = perm.begin();it_ != perm.end();++it_)

        {   for(vector<stages>::iterator st = q.begin();st != q.end();++st )

            {  it1 = st->begin();

                if(*(it_ +1) != *(perm.end()))

                    if( (*it_ == *it1) && (*(it_ +1) == *(it1 +1)))

                        {   temp.clear();

                            it2 = it1+2;

                    while(it2 != st->end())

                        {    temp.push_back(*it2);

                             ++it2;
                        }

                            qTemp.push_back(temp);
                    }
                }
            }

    return qTemp;

    }
```

```
// function to input the stage setup time data according to the number of pro
ducts
    void inputQ(vector <stages> & q,int pLength, int length){
    float t;
    vector<float> temp;
    q.clear();
    for(int i=1;i<=pLength;i++){
        for(int j=1;j<= pLength;j++){
            if (i!= j){
            temp.clear();
            temp.push_back(i);
            temp.push_back(j);
            cout<<endl<<"Enter Stage Setup Time between the products ";
            cout<<"P"<< i << "-P" << j <<endl;
            cout<<"-------------------------------------------------"<<endl<
<endl;
        for(int k=1;k<=length;k++){
                        {   cout<<"Stage("<<k<<")= ";
                            cin>>t;
                //          t=(rand()%5)+1;
                //          cout<<t;
                            cout<<endl;
                            }
                            temp.push_back(t);
                        }
                            q.push_back(temp);
                }
            }
        }
    }
    void main()         // main Program
    {
    back:
```

```
    srand((unsigned)time(0));

    Stopwatch s;        // call stopwatch function

    system("cls");

    char ans;           // character varaible to accept user input (y/n)


    cout<<endl;

    cout<<"*************************************************************"<<e
ndl;

    cout<<"* Scheduling Multiproduct Batch Process on Makespan Criteria *"<<e
ndl;

    cout<<"*                                                            *"<<e
ndl;

    cout<<"* Compiler: Microsoft Visual C++ ver 6.0                     *"<<e
ndl;

    cout<<"*                                                            *"<<e
ndl;

    cout<<"* Platform: Pentium IV 2.80GHz, Windows XP                   *"<<e
ndl;

    cout<<"*                                                            *"<<e
ndl;

    cout<<"* Programmer: Amir Shafeeq                                   *"<<e
ndl;

    cout<<"*                                                            *"<<e
ndl;

    cout<<"* Date: March 2008                                           *"<<e
ndl;

    cout<<"*************************************************************"<<e
ndl<<endl<<endl<<endl;

    cout<<"Enter number of Products (Minimum = 2 , Maximum = 100)"<<endl<<end
l;

    cin>>p;              // input number of products in the given batch proces
s

    if (p> 100 || p < 2)

        goto back;

    cout<<endl;

    cout<<"Enter number of stages   (Minimum = 2 , Maximum = 100) "<<endl<<en
dl;

    cin>>g;          // input number of stages in the given batch process

    if (g> 100 || g < 2)
```

```
        goto back;

    for (i=1;i<=p;i++)
        {
            M[i][0]=i;          // assign number 1,2,3..to rows of product re
cipe
            cout<<endl<<endl;
            cout<<"Enter Processing Time of Product P"<<i<<""<<endl;
            cout<<"-----------------------------------"<<endl<<endl;
            for (j=1;j<=g;j++)
            {   cout<<"Stage("<<j<<")= ";
                cin>>M[i][j];   // input processing times of products
            //  M[i][j] = (rand()%50)+1;
            //  cout<<M[i][j];
                cout<<endl;
            }
        }

    cout<<endl;
    cout<< "Do you wish to input the transfer and setup time data ? (y/n)"<<e
ndl<<endl;
    cin >> ans;
    getchar();
    if (ans == 'n' || ans == 'N') goto loop;
        system("cls");
        for (i=1;i<=p;i++)
        {   cout<<"Enter Transfer Time of Product P"<<i<<""<<endl;
            cout<<"-----------------------------------"<<endl<<endl;
            for (j=g+1;j<=(g*2)+1;j++)
            {   cout<<"Stage("<<j-g-1<<")= ";
                cin>>M[i][j];       // input transfer time of products
        //      M[i][j] = (rand()%5)+1;
        //      cout<<M[i][j];
                cout<<endl;
            }
```

```
                cout<<endl<<endl;
        }
       inputQ(Q,p,g);              // call stored stage setup time data
loop:

// storing the initial data in an array to restore first production sequence
      for (i=1;i<=p;i++)
         {   for (j=0;j<= (g*3)+1;j++)
             MM[i][j] = M[i][j];
         }
again:

    system("cls");
//  myfile.open("example.txt"); // create file to store program output
    cout<< "Select Transfer Policy"<<endl;
    cout<< "======================"<<endl;
    cout<< "1. Zero Wait (ZW)"<<endl<<endl;
    cout<< "2. No Intermediate Storage (NIS)"<<endl<<endl;
    cout<< "3. Unlimited Intermediate Storage (UIS)"<<endl<<endl;
    cout<< "4. Finite Intermediate Storage (FIS)"<<endl<<endl;
    cout<< "5. Mixed Intermediate Storage (MIS -> NIS='1->m-2'/UIS='m-1'"<<en
dl;
    cout<< "============================================================"<<en
dl<<endl;
    cin>>policy;               // input transfer policy number (1/2/3/4/5)
    cout<<endl;
    if(policy > 5 || policy <= 0)
        {   cout<<endl<<endl<<"Error Entering Policy"<<endl<<endl;
            system ("pause");
            goto loop;
        }
    if (policy==4)
        {   cout<<endl<<endl;
            cout<< "Do you wish to input the Intermediate Storage setup time
data ? (y/n)"<<endl<<endl;
```

```
            cin >> ans;
            getchar();
    if (ans == 'y' || ans == 'Y')
        {   system ("cls");
        for (i=1;i<=p;i++)
            {
                cout<<"Enter Intermediate Storage Setup time of Product P"<<i
<<endl;
                cout<<"-----------------------------------------------------"
<<endl<<endl;
                for (j=(g*2)+2;j<=(g*3)+1;j++)
                {   cout<<"Stage("<<j-((g*2)+1)<<")= ";
                cin>>M[i][j];   // input setup time for temporary storage
        //      M[i][j] = (rand()%5)+1;
        //      cout<<M[i][j];
                    cout<<endl;
                }
            cout<<endl<<endl;
            }
        }
  }
    system("cls");
    cout<< "Select Type Of Enumeration"<<endl<<endl;
    cout<<"=========================="<<endl<<endl;
    cout<<" 1. Total Enumeration"<<endl<<endl;
    cout<<" 2. Partial  Enumeration"<<endl<<endl;
    cout<<"=========================="<<endl<<endl;
    cin >> method;
    cout<<endl<<endl;

// Heuristic procedure for partial enumeration
//  for (int r = 0; r <MergeK; r++)
//  MergeIndex[r] =0;MergeK=0;
```

```
    if (method == 2)
{
            float sumlast=0.0;
            for (i=1;i<=p;i++)
            sumlast=M[i][g]+M[i][g*2]+M[i][(g*2)+1]+sumlast;
            int sumrow[100][2]={0};
            for (i=1;i<=p;i++)
            {   for (j=1;j<g;j++)
                {
                    sumrow[i][0]+=M[i][j]+M[i][g+j];
                }
                    sumrow[i][0]+=sumlast; // sum of rows
      //            cout<<sumrow[i][0]<<"   ";
                    sumrow[i][1]= i;       // position of rows
      //            cout<<sumrow[i][1]<<endl;
            }
       // assign values of sumrow to MergeIndex
      //   cout<<"Before Sorting"<<endl;
            for (i=1;i<=p;i++)
                for (j=0;j<2;j++){
                    MergeIndex1[i][j] = sumrow[i][j];
      //            cout<<MergeIndex1[i][j]<<endl;
                }
     //  Calling sortArray function will sort MergeIndex1 global array

                sortArray(p);
     //printing the values of mergeindex array
      //   cout<<"After sorting the array";
      //   for (i=1;i<=p;i++){
      //       cout<<endl<<MergeIndex1[i][0]<<"    ";
      //       cout<<MergeIndex1[i][1]<<endl;
      //   }
            //  Minimum value of first column
             int colvalues[100][2];
             for(i=1;i<=p;i++){
                colvalues[i][0]= M[i][1]+M[i][g+1];//first column of 2 matri
ces
                colvalues[i][1]= i;// Position of columns
               }
                // assign values of col to MergeIndex2
     //   cout<<"Before Sorting"<<endl;
            for (i=1;i<=p;i++)
               for (j=0;j<2;j++){
                   MergeIndex2[i][j] = colvalues[i][j];
     //            cout<<MergeIndex2[i][j]<<endl;
               }
    //  Calling sortArray function will sort MergeIndex1 global array

               sortArray2(p);
    //printing the values of mergeindex array
     //  cout<<"After sorting the array";
     //  for (i=1;i<=p;i++){
     //      cout<<endl<<MergeIndex2[i][0]<<"   ";
     //      cout<<MergeIndex2[i][1]<<endl;
     //  }
         //cout<<"counter initial  " <<counter<<endl;
         Minrow1 = MergeIndex1[1][0];
         while (Minrow1 == MergeIndex1[rows][0]){
             MergeIndex[counter3]=MergeIndex1[rows][1];
             rows++; counter3++;
         }
         //cout<<"counter 1 t" <<counter<<endl;
         Mincol1 = MergeIndex2[1][0];
         while (Mincol1 == MergeIndex2[cols][0]){
             MergeIndex[counter3]=MergeIndex2[cols][1];
             counter3++;cols++;
         }
             //cout<<"counter 2 t" <<counter<<endl;
             MergeK = counter3;
             //cout<<"After Merging the MI1 and MI2"<<endl;
```

```
                //for(int z=1; z<counter;z++)
                //  cout<<MergeIndex[z]<<"   ";
        }

// select the transfer policy and execute its algorithm
    switch (policy)
    {
    case ZW:
        fp =  zmycode;
    break;
    case NI:
        fp = nmycode;
    break;
    case UI:
        fp = umycode;
    break;
    case FI:
        fp = cmycode;
    break;
    case MI:
        fp = mmycode;
    break;

    }

backhere:
  s.reset();        // CPU timer starts
  myfile.open("example.txt");   // create file to store program output
  prog(fp);             // get the output from the executed algorithm
  myfile<<"No. of Possible Production Sequences = "<<temp<<endl<<endl;
  if (method == 1) goto next;
  myfile<<"No. of Partial Production Sequences = "<<counter2<<endl<<endl;
next:
  myfile<<"No. of Production Sequences with Minimum Makespan = "<<kk<<endl<<e
ndl;

  myfile<<"Optimum Production Sequences with Minimum Makespan"<<endl<<endl;

// display production sequences with minimum makespan only


   for(int uu=1;uu<=kk;uu++)

  {   for (int u=1;u<=p;u++)

          myfile<<"P"<<MN[u][uu];

          myfile<<"    = "<<min<<" hours"<<endl;

  }
    float t = s.elapsedTime();      // CPU timer stops

    myfile<<endl;

    myfile << "CPU Time = " << t << " seconds" << endl<<endl;

    myfile.close();                 // close the output file

    system("pause");

    cout<<endl<<endl<<endl;

    if (method==2)

    {

    char nextmin;

    cout<< "Do you wish to run the program again using next minimum value ? (
y/n)"<<endl<<endl;

    cin >> nextmin;

    getchar();

    cout<<endl<<endl<<endl;


    if (nextmin == 'y' || nextmin == 'Y')


    {           if (counter2==temp) goto nomore;


                 for (i=1;i<=p;i++)      // restore the initial data

                 {        for (j=0;j<= (g*3)+1;j++)

                          M[i][j] = MM[i][j];
                 }


                 if (rows <= p){

                 Minrow1 = MergeIndex1[rows][0];
```

```
                while (Minrow1 == MergeIndex1[rows][0]){
                    MergeIndex[counter3]=MergeIndex1[rows][1];
                    rows++; counter3++;
                    }
            //  cout<<"counter 1 t" <<counter<<endl;
            }
            if (cols <=g){
                Mincol1 = MergeIndex2[cols][0];
                while (Mincol1 == MergeIndex2[cols][0]){
                    MergeIndex[counter3]=MergeIndex2[cols][1];
                    counter3++;cols++;
                }
            }
        //      cout<<"counter 2 t" <<counter<<endl;
                MergeK = counter3;
        //      cout<<"After Merging the MI1 and MI2"<<endl;
        //  for(z1=1; z1<counter;z1++)
        //      cout<<MergeIndex[z1]<<"    ";
    goto backhere;

nomore:
    cout<<"No more minimum value possible, reached to last product in the seq
uence"<<endl;
    cout<<"=============================================================
====="<<endl<<endl<<endl;
    }
}
    cout<<endl<<endl<<endl;
    cout<< "Do you wish to run the program again using same data ? (y/n)"<<en
dl<<endl;
    cin >> ans;
    getchar();
    if (ans == 'y' || ans == 'Y')
        {
                for (i=1;i<=p;i++)      // restore the initial data
                {       for (j=0;j<= (g*3)+1;j++)
                        M[i][j] = MM[i][j];
                }
                for(int delr = 0; delr<MergeK;delr++) // initilize all mergei
ndex arrays & variables
                    MergeIndex[delr] = 0;
                for(int dr=0;dr<100;dr++)
                    for(int ds=0;ds<2;ds++){
                        MergeIndex1[dr][ds] =0;
                        MergeIndex2[dr][ds] =0;
                    }
                counter3=1,
                rows = 1,
                cols = 1,
                MergeK = 0;
                goto again;
        }

    cout<<endl<<endl<<endl;
        cout<< "Do you wish to run the program again using new data ? (y/n)"<
<endl<<endl;
        cin >> ans;
        getchar();
        if (ans == 'y' || ans == 'Y')
        {   for(int delr = 0; delr<MergeK;delr++)    // initilize all mergeind
ex arrays & variables
                        MergeIndex[delr] = 0;
                for(int dr=0;dr<100;dr++)
                    for(int ds=0;ds<2;ds++){
                        MergeIndex1[dr][ds] =0;
                        MergeIndex2[dr][ds] =0;
                    }

                counter3=1,
                rows = 1,
                cols = 1,
                MergeK = 0;

            goto back;
        }
        cout<<endl;
    }
    int prog(float (*fp)())
    {
    ca.clear();
    t.clear();
    ArrayToVector();
    ArrayToVectornew();
    t = displaypermuQ(Q,permt);
    VectorToArraynew(t,Mt);
    cout<<endl;
    if (policy == 1)
        myfile<<"Number of Products = "<<p<<"    Number of Stages = "<<g<<"
Transfer Policy = ZW";
    if (policy == 2)
        myfile<<"Number of Products = "<<p<<"    Number of Stages = "<<g<<"
Transfer Policy = NIS";
    if (policy == 3)
        myfile<<"Number of Products = "<<p<<"    Number of Stages = "<<g<<"
Transfer Policy = UIS";
    if (policy == 4)
        myfile<<"Number of Products = "<<p<<"    Number of Stages = "<<g<<"
Transfer Policy = FIS";
    if (policy == 5)
        myfile<<"Number of Products = "<<p<<"    Number of Stages = "<<g<<"
Transfer Policy = MIS";
        myfile<<endl<<"_____________________________________________________
________________________"<<endl<<endl;

        if (method == 2)
        {
            int counter=0;
            min=0, kk=1;
            bool firstflag = false;
            for (int r = 0; r <MergeK; r++)
            if (MergeIndex[r] == M[1][0])
            firstflag = true;
            if(firstflag)
        {
//    for (int u=1;u<=p;u++)
//    myfile<<"P"<<M[u][0];
            counter++;
            fp();
//  myfile<<" =  "<<MAKE<<" hours"<<endl<<endl<<endl;
           min=MAKE;
    for (int u=1;u<=p;u++)
        MN[u][1]=M[u][0];
        }

        for(int i=1;i<=factor(p)-1;++i)
        {   next_permutation(ca.begin(),ca.end());
            VectorToArray();
                bool flag = false;
                for (int r = 0; r <MergeK; r++)
                    if (MergeIndex[r] == M[1][0])
                        flag = true;
                        if(flag)
            {
            counter++;
            ArrayToVectornew();
            t = displaypermuQ(Q,permt);
            VectorToArraynew(t,Mt);
            cout<<endl;
  //        for (int u=1;u<=p;u++)
  //        myfile<<"P"<<M[u][0];
            fp();
  //        myfile<<" =  "<<MAKE<<" hours"<<endl<<endl<<endl;

                if (min==0)
                {   min=MAKE;
                    kk = 1;
                    for (int u=1;u<=p;u++)
                    MN[u][kk]=M[u][0];
                }
// optimal solution screening based on minimum makespan criteria by making co
mparison of makespan of each production sequence resulted from permutation of
 the product recipe
                 else if (min>MAKE)
                        {   min=MAKE;
                            kk = 1;
                            for (int u=1;u<=p;u++)
                            MN[u][kk]=M[u][0];
                         }
                else if(min==MAKE)
                        {   kk=kk+1;
                            for (int u=1;u<=p;u++)
                            MN[u][kk]=M[u][0];

                        }
                }
        }
         counter2=counter;
        }

      if (method == 1)
      {  min=0, kk=1;
//    for (int u=1;u<=p;u++)
//   myfile<<"P"<<M[u][0];

             fp();

//  myfile<<" =  "<<MAKE<<" hours"<<endl<<endl<<endl;

           min=MAKE;

    for (int u=1;u<=p;u++)

        MN[u][1]=M[u][0];


        for(int i=1;i<=factor(p)-1;++i)

        {    next_permutation(ca.begin(),ca.end());

             VectorToArray();

             ArrayToVectornew();

             t = displaypermuQ(Q,permt);

             VectorToArraynew(t,Mt);

             cout<<endl;

  //         for (int u=1;u<=p;u++)

  //         myfile<<"P"<<M[u][0];

             fp();

  //        myfile<<" =  "<<MAKE<<" hours"<<endl<<endl<<endl;

// optimal solution screening based on minimum makespan criteria by making co
mparison of makespan of each production sequence resulted from permutation of
 the product recipe

                   if (min>MAKE)

                           {    min=MAKE;

                                kk = 1;

                                for (int u=1;u<=p;u++)

                                MN[u][kk]=M[u][0];

                            }

                  else if(min==MAKE)

                           {    kk=kk+1;

                                for (int u=1;u<=p;u++)

                                MN[u][kk]=M[u][0];
```

```
                              }
                      }
        }
           return 0;
}

//  Zero Wait (ZW) Algorithm

    float zmycode()
        {   float W=0.0,Y=0.0,Z=0.0,X=0.0,MAX=0.0,MAXIM=0.0;

        for (i=1;i<p;i++)                     // equation (4.1a)
            {   V[i][2]=(M[i+1][1]+M[i+1][g+1])-(M[i][2]+M[i][g+3]);
                if (V[i][2]<0.0)
                    V[i][2]=0.0;
            }
        for (i=1;i<p;i++)                     // equation (4.2a)
            {   for (j=2;j<g;j++)
                {   V[i][j+1]=(V[i][j]+(M[i+1][j]+M[i+1][j+g]))-(M[i][j+1]+M[
i][j+g+2]);
                    if (V[i][j+1]<0.0)
                        V[i][j+1]=0.0;
                }
            }
        for (i=1;i<p;i++)                     // equation (4.3a)
            {   for (j=g-1;j>=1;j--)
                {   V[i][j]=(V[i][j+1]-(M[i+1][j]+M[i+1][j+g]))+(M[i][j+1]+M[
i][j+g+2]);
                    if (V[i][j]<0.0)
                        V[i][j]=0.0;
                }
            }
// setup time (required) comparison with existing inter-stage idle times
        for (i=1;i<p;i++)
            {   for (j=1;j<=g;j++)
                { CT[i][j]=Mt[i][j-1] - V[i][j];
                  MAXIM=CT[i][j];
                if (MAXIM>MAX)
                MAX=MAXIM;
            }
        for (j=1;j<=g;j++)
            V[i][j]=V[i][j]+MAX;
            MAX=0.0;
        }
        for (i=2;i<=p;i++)
            {   W= W+M[i][g]+M[i][((2*g)+1)];
            }
        for (i=2;i<=p;i++)
            {   X= X+M[i][(2*g)];
            }
        for (j=1;j<=g;j++)
            {   Y=Y+M[1][j]+M[1][g+j+1];
            }

        for (i=1;i<p;i++)
            {   Z=Z+V[i][g];
            }
            MAKE=W+X+Y+Z+M[1][g+1];         // equation (4.4b)
            data();
            return 0;
    }

//  No Intermediate Storage (NIS) Algorithm

float nmycode()
    {   float W=0.0,Y=0.0,Z=0.0,X=0.0;
// addition of transfer times to the processing times in accordance with NIS
algorithm
        for (i=2;i<=p;i++)
            M[i][1]=M[i][1]+Mt[i-1][0];
        for (i=1;i<p;i++)
        {   V[i][1]=0.0;
            I[i][g]=0.0;
        }
        for (j=1;j<g;j++)                        // equation (4.5a)
            {   V[1][j+1]=(V[1][j]+M[2][j]+M[2][g+j])-(M[1][j+1]+M[1][g+j+2])
;
                if (V[1][j+1]<Mt[1][j])
                    {   V[1][j+1]=Mt[1][j];
                        I[1][j]= (V[1][j+1]+M[1][j+1]+M[1][g+j+2])- (V[1][j]+
M[2][j]+M[2][g+j]);
                    }
                else
                    {   V[1][j+1]=V[1][j+1];
                        I[1][j] = 0.0;
                    }
            }
        for (i=2;i<p;i++)                        // equation (4.6a)
        {
            for (j=1;j<g;j++)
            {
                V[i][j+1]=(V[i][j]+M[i+1][j]+M[i+1][g+j])-(M[i][j+1]+I[i-1][j
+1]+M[i][g+j+2]);

                if (V[i][j+1]<Mt[i][j])
                    {   V[i][j+1]=Mt[i][j];
                        I[i][j]= (V[i][j+1]+M[i][j+1]+I[i-1][j+1]+M[i][g+j+2]
)- (V[i][j]+M[i+1][j]+M[i+1][g+j]);
                    }
            else
                {   V[1][j+1]=V[1][j+1];
                    I[i][j] = 0.0;
                }
            }
        }
        for (i=2;i<=p;i++)
            {   W= W+M[i][g]+M[i][((2*g)+1)];
            }
        for (i=2;i<=p;i++)
            {   X= X+M[i][(2*g)];
            }

        for (j=1;j<=g;j++)
            {   Y=Y+M[1][j]+M[1][g+j+1];
            }

        for (i=1;i<p;i++)
            {   Z=Z+V[i][g];
            }

        MAKE=W+Y+X+Z+M[1][g+1];             // equation (4.4c)
// exclusion of transfer times from the proessing times to displaythe process
ing times only
        for (i=2;i<=p;i++)
            M[i][1]=M[i][1]-Mt[i-1][0];
        for (i=1;i<p;i++)
            V[i][1]=V[i][1]+Mt[i][0];
        data();
        ntank();
        return 0;
    }

// function to display holding time inside stage for NIS

    float ntank()
        {   for (i=1;i<p;i++)
            {   for (j=1;j<g;j++)
    //              myfile<<"I" << i << "," << j << "=" <<I[i][j]<<"    ";
            ;}
    //  myfile<<endl<<"-------------------------------------------------------
------------------------"<<endl;
        return 0;
    }
```

```
//  Unlimited Intermediate Storage (UIS) Algorithm

    float umycode()

        {   float R=0.0,Y=0.0,Z=0.0,X=0.0;

// addition of transfer times to the proessing times in accordance with UIS a
lgorithm

        uiscis();

        for (i=1;i<p;i++)

            V[i][1]=0.0;

        for (j=2;j<=g;j++)          // equation (4.7) & equation (5.3)

            {   WW[1][j-1]=0.0;

                V[1][j]= (V[1][j-1]+M[2][j-1])-(M[1][j]);

                if (V[1][j]<Mt[1][j-1])

                    {   V[1][j]=Mt[1][j-1];

                        W[1][j-1]= (V[1][j]+M[1][j])-(V[1][j-1]+M[2][j-1]+M[2
][g+j]);

                        if (W[1][j-1]<=0.0)

                        {
                            V[1][j]=V[1][j]+(-W[1][j-1]);

                            WW[1][j-1]= 1;

                            W[1][j-1]=0.0;
                        }
                    }

                else

                    {   V[1][j]=V[1][j];

                        W[1][j-1] = 0.0;
                    }
            }

        for (i=2;i<p;i++)          // equation (4.8b) & equation (5.4)

            {   for (j=2;j<=g;j++)

                {
                    float A=0.0,B=0.0,C=0.0,D=0.0,AA=0.0,CC=0.0;

                    for (k=2;k<i+2;k++)

                        A=A+M[k][j-1];
                for (k=2;k<i+1;k++)
                    AA=AA+M[k][g+j];
                for (k=1;k<i+1;k++)
                    B=B+V[k][j-1];
                for (k=1;k<i+1;k++)
                    C=C+M[k][j];
                for (k=2;k<i+1;k++)
                    CC=CC+M[k][g+j+1];
                for (k=1;k<i;k++)
                    D=D+V[k][j];
                    WW[i][j-1]=0.0;
                    V[i][j] = (A+B+AA)-(C+D+CC);
                if (V[i][j]<Mt[i][j-1])
                    {   V[i][j]=Mt[i][j-1];
                        W[i][j-1]= (V[i][j]+C+D+CC)-(A+B+AA+M[i+1][g+j]);
                        if (W[i][j-1]<=0.0)
                            {   V[i][j]=V[i][j]+(-W[i][j-1]);
                                WW[i][j-1]= 1;
                                W[i][j-1]=0.0;
                            }
                    }
                else
                    {   V[i][j]=V[i][j];
                        W[i][j-1] = 0.0;
                    }
                }
        }
    for (i=2;i<=p;i++)
        {   R= R+M[i][g];
        }
    for (j=1;j<=g;j++)
        {   Y=Y+M[1][j];
        }
        for (i=2;i<=p;i++)
            {   X= X+M[i][(2*g+1)];
            }
        for (i=1;i<p;i++)
            {   Z=Z+V[i][g];
            }

        MAKE=R+Y+Z+X;                // equation (4.4d)
// exclusion of transfer times from the proessing times to display the proces
sing times only
        uiscis2();
        for (i=1;i<p;i++)
            V[i][1]=V[i][1]+Mt[i][0];
        data();
        utank();
        return 0;
    }

// function to display waiting time inside temporary storage for UIS

    float utank()
    {  for (i=1;i<p;i++)
        {   for (j=1;j<g;j++)
            { if (WW[i][j] == 1)
    //              myfile<<"W" << i << "," << j << "= pass"<<"    ";
            ;  else
    //              myfile<<"W" << i << "," << j << "=" <<W[i][j]<<"    ";
    ;       }
        }
    //  myfile<<endl<<"----------------------------------------------------
------------------------"<<endl;
        return 0;
    }


//  Finite Intermediate Storage (FIS) Algorithm
```

```
    float cmycode()

    {   float R=0.0,Y=0.0,Z=0.0,X=0.0;

// addition of transfer times to the proessing times in accordance with FIS a
lgorithm

        uiscis();

        for (i=1;i<p;i++)

            V[i][1]=0.0;

        for (j=1;j<=g;j++)

            I[1][j]=0.0;

        for (i=2;i<p;i++)

            I[i][g]=0.0;


        for (j=2;j<=g;j++)       // equation (4.9) & equation (5.5)

            {   WW[1][j-1]=0.0;

                V[1][j]=(V[1][j-1]+M[2][j-1])-M[1][j];

                if (V[1][j]<Mt[1][j-1])

                {   V[1][j]=Mt[1][j-1];

                    W[1][j-1]= (V[1][j]+M[1][j])-(V[1][j-1]+M[2][j-1]+M[2][g+
j]);

                    if (W[1][j-1]<=0.0)

                    {
                        V[1][j]=V[1][j]+(-W[1][j-1]);

                        WW[1][j-1]= 1;

                        W[1][j-1]=0.0;
                    }
                }

            else

                {   V[1][j]=V[1][j];

                    W[1][j-1] = 0.0;
                }
        }

    for (i=2;i<p;i++)       // equation (4.10a) & equation (5.6)

        {   for (j=2;j<=g;j++)

            {   float A=0.0,B=0.0,C=0.0,D=0.0,E=0.0,F=0.0,AA=0.0,DD=0.0;

                if (M[i][(g*2)+j]<=((M[i+1][j-1]+V[i][j-1])-(W[i-1][j-1]+M[i]
[g+j])))
```

```
                    M[i][(g*2)+j]=0.0;
                if (W[i-1][j-1]>0)
                    {   if ((M[i+1][j-1]+V[i][j-1])<=(W[i-1][j-1]+M[i][g+j]+M
[i][(g*2)+j]))
                            I[i][j-1]=(W[i-1][j-1]+M[i][g+j]+M[i][(g*2)+j])-(
M[i+1][j-1]+V[i][j-1]);
                        else
                            I[i][j-1]=0.0;
                    }
                        else
                            I[i][j-1]=0.0;
        for (k=2;k<i+2;k++)
            A=A+M[k][j-1];
        for (k=2;k<i+1;k++)
            AA=AA+M[k][g+j];
        for (k=1;k<i+1;k++)
            B=B+V[k][j-1];
        for (k=1;k<i+1;k++)
            C=C+I[k][j-1];
        for (k=1;k<i+1;k++)
            D=D+M[k][j];
        for (k=2;k<i+1;k++)
            DD=DD+M[k][g+j+1];
        for (k=1;k<i;k++)
            E=E+V[k][j];
        for (k=1;k<i;k++)
            F=F+I[k][j];
            WW[i][j-1]= 0.0;
            V[i][j] = (A+B+C+AA)-(D+E+F+DD);
            if (V[i][j]<Mt[i][j-1])
                {   V[i][j]=Mt[i][j-1];
                    W[i][j-1]= (V[i][j]+D+E+F+DD)- ((A+B+C+AA)+M[i+1][g+j]);
                if (W[i][j-1]<=0.0)
                    {   V[i][j]=V[i][j]+(-W[i][j-1]);
                        WW[i][j-1]= 1;
                        W[i][j-1]=0.0;
                    }
                }
            else
                {   V[i][j]=V[i][j];
                    W[i][j-1] = 0.0;
                }
            }
        }
        for (i=2;i<=p;i++)
            {   R= R+M[i][g];
            }
        for (j=1;j<=g;j++)
            {   Y=Y+M[1][j];
            }
        for (i=2;i<=p;i++)
            {   X= X+M[i][(2*g+1)];
            }
        for (i=1;i<p;i++)
            {   Z=Z+V[i][g];
            }
        MAKE=R+Y+X+Z;           // equation (4.4d)

// exclusion of transfer times from the proessing times to display the proces
sing times only

        uiscis2();
        for (i=1;i<p;i++)
            V[i][1]=V[i][1]+Mt[i][0];
        data();
        ctank();
        return 0;
    }

// function to display waiting time inside temporary storage and holding time
 inside stage for FIS

    float ctank()

    {   for (i=1;i<p;i++)

            {
                for (j=1;j<g;j++)

                {   if (WW[i][j] == 1)

        //                  myfile<<"W" << i << "," << j << "= pass"<<"    ";

            ;           else

        //                  myfile<<"W" << i << "," << j << "=" <<W[i][j]<<"    ";
        ;       }
            }
        //                  myfile<<endl<<"---------------------------------------
---------------------------------------"<<endl;

        for (i=1;i<p;i++)

            {
                for (j=1;j<g;j++)

        //                  myfile<<"I" << i << "," << j << "=" <<I[i][j]<<"    ";
        ;   }
        //                  myfile<<endl<<"---------------------------------------
---------------------------------------"<<endl;

        return 0;

    }

// function to display inter-stage idle times

    float data()

    {   //myfile<<endl<<"----------------------------------------------------
--------------------------"<<endl;


        for (i=1;i<p;i++)

            {
                for (j=1;j<=g;j++)

        //              myfile<<"H" << i << "," << j << "="<<V[i][j]<<"    ";
        ;   }
        //              myfile<<endl<<"-------------------------------------------
-----------------------------------"<<endl;
```

```
        return 0;

    }

// function to add transfer times to the processing times in accordance with
algorithm of UIS and FIS for the purpose of calculation

    float uiscis()

    {   for (i=2;i<=p;i++)

            M[i][1]=M[i][1]+Mt[i-1][0]+M[i][g+1];

            M[1][1]=M[1][1]+M[1][g+1];

        for (j=1;j<=g;j++)

            M[1][j]=M[1][j]+M[1][g+j+1];


        for (j=2;j<g;j++)

            {   for (i=2;i<=p;i++)

                    M[i][j]=M[i][j]+M[i][g+j];
            }

        for (j=g;j<=g;j++)

            {   for (i=2;i<=p;i++)

                    M[i][j]=M[i][j]+M[i][g*2];
            }

        return 0;
    }

// function to exclude transfer times from the processing times to display on
ly the processing times in UIS and FIS

    float uiscis2()

    {   for (i=2;i<=p;i++)

            M[i][1]=M[i][1]-(Mt[i-1][0]+M[i][g+1]);

            M[1][1]=M[1][1]-M[1][g+1];

        for (j=1;j<=g;j++)

            M[1][j]=M[1][j]-M[1][g+j+1];


        for (j=2;j<g;j++)

            {   for (i=2;i<=p;i++)

                    M[i][j]=M[i][j]-M[i][g+j];
            }
        for (j=g;j<=g;j++)
            {   for (i=2;i<=p;i++)
                    M[i][j]=M[i][j]-M[i][g*2];
            }
        return 0;
    }


//  Mixed Intermediate Storage (MIS) Algorithm


float mmycode()
        {   float R=0.0,Y=0.0,Z=0.0,X=0.0;
// addition of transfer times to the processing times in accordance with NIS
algorithm
        for (i=2;i<=p;i++)
            M[i][1]=M[i][1]+Mt[i-1][0];
        for (i=1;i<p;i++)
        {   V[i][1]=0.0;
            I[i][g-1]=0.0;
        }
        for (j=1;j<g-1;j++)                     // equation (4.5c) & equation (5.
1a)
            {   V[1][j+1]=(V[1][j]+M[2][j]+M[2][g+j])-(M[1][j+1]+M[1][g+j+2])
;
                if (V[1][j+1]<Mt[1][j])
                    {   V[1][j+1]=Mt[1][j];
                        I[1][j]= (V[1][j+1]+M[1][j+1]+M[1][g+j+2])- (V[1][j]+
M[2][j]+M[2][g+j]);
                    }
                else
                    {   V[1][j+1]=V[1][j+1];
                        I[1][j] = 0.0;
                    }
            }
        for (i=2;i<p;i++)                       // equation (4.6c) & equation (5.
2a)
```

```
        {
            for (j=1;j<g-1;j++)

            {
                V[i][j+1]=(V[i][j]+M[i+1][j]+M[i+1][g+j])-(M[i][j+1]+I[i-1][j
+1]+M[i][g+j+2]);

                if (V[i][j+1]<Mt[i][j])

                    {   V[i][j+1]=Mt[i][j];

                        I[i][j]= (V[i][j+1]+M[i][j+1]+I[i-1][j+1]+M[i][g+j+2]
)- (V[i][j]+M[i+1][j]+M[i+1][g+j]);
                    }

            else

                {   V[1][j+1]=V[1][j+1];

                    I[i][j] = 0.0;

                }
            }
        }

// exclusion of transfer times from the proessing times to displaythe process
ing times only

        for (i=2;i<=p;i++)

            M[i][1]=M[i][1]-Mt[i-1][0];

        for (i=1;i<p;i++)

            V[i][1]=V[i][1]+Mt[i][0];


//  Unlimited Intermediate Storage (UIS) Algorithm


// addition of transfer times to the proessing times in accordance with UIS a
lgorithm

        uiscis();

                                                      // equation (4.7a) & eq
uation (5.3a)
                j=g;

                WW[1][j-1]=0.0;

                V[1][j]= (V[1][j-1]+M[2][j-1])-(M[1][j]);


                if (V[1][j]<Mt[1][j-1])

                    {   V[1][j]=Mt[1][j-1];

                        W[1][j-1]= (V[1][j]+M[1][j])-(V[1][j-1]+M[2][j-1]+M[2
][g+j]);
                        if (W[1][j-1]<=0.0)
                        {
                            V[1][j]=V[1][j]+(-W[1][j-1]);
                            WW[1][j-1]= 1;
                            W[1][j-1]=0.0;
                        }
                    }
                else
                    {   V[1][j]=V[1][j];
                        W[1][j-1] = 0.0;
                    }

        for (i=2;i<p;i++)       // equation (4.8c) & equation (5.4a)
            {       j=g;
                    float A=0.0,B=0.0,C=0.0,D=0.0,AA=0.0,CC=0.0;
                    for (k=2;k<i+2;k++)
                        A=A+M[k][j-1];
                    for (k=2;k<i+1;k++)
                        AA=AA+M[k][g+j];
                    for (k=1;k<i+1;k++)
                        B=B+V[k][j-1];
                    for (k=1;k<i+1;k++)
                        C=C+M[k][j];
                    for (k=2;k<i+1;k++)
                        CC=CC+M[k][g+j+1];
                    for (k=1;k<i;k++)
                        D=D+V[k][j];
                        WW[i][j-1]=0.0;
                        V[i][j] = (A+B+AA)-(C+D+CC);
                    if (V[i][j]<Mt[i][j-1])
                        {   V[i][j]=Mt[i][j-1];
                            W[i][j-1]= (V[i][j]+C+D+CC)-(A+B+AA+M[i+1][g+j]);
                            if (W[i][j-1]<=0.0)
```

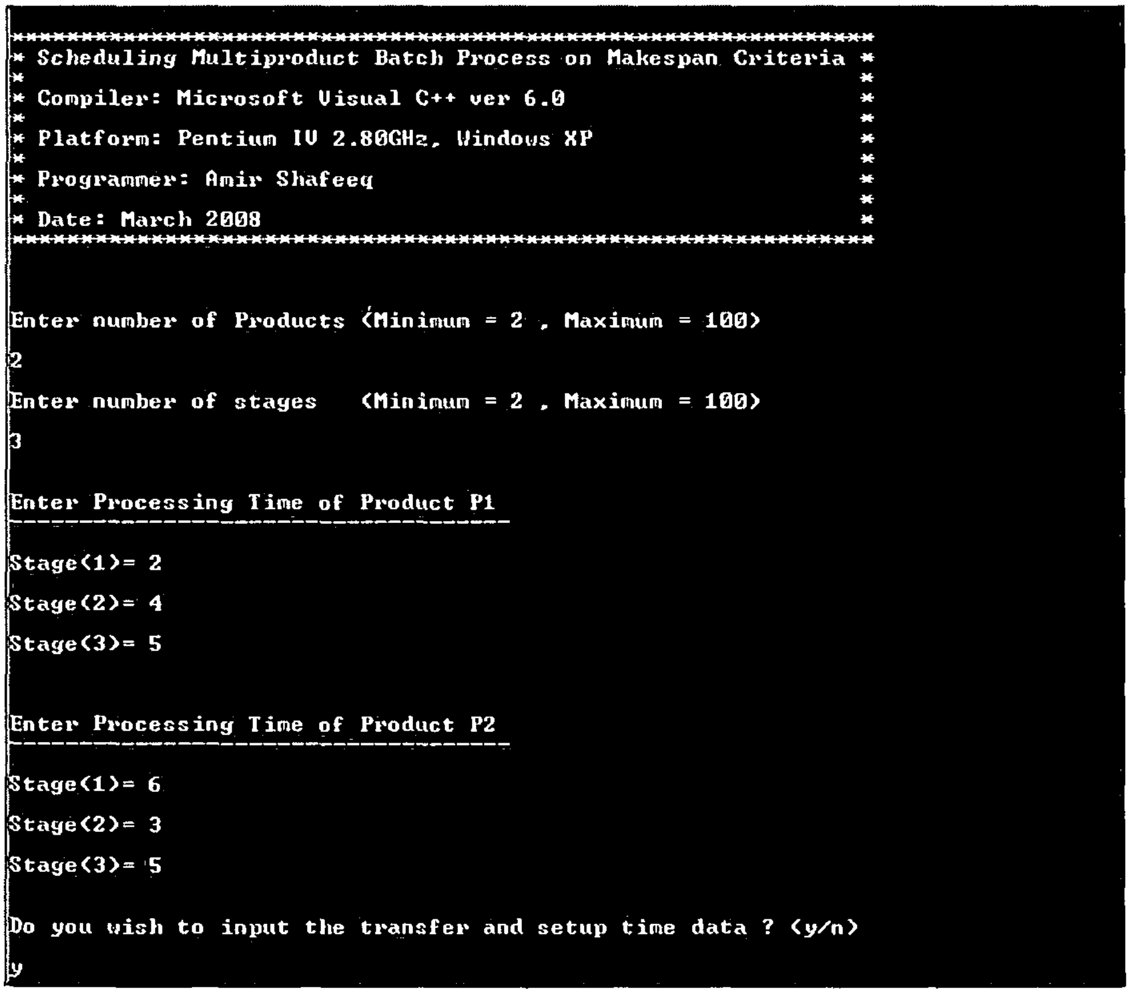
```
                                        {   V[i][j]=V[i][j]+(-W[i][j-1]);
                                            WW[i][j-1]= 1;
                                            W[i][j-1]=0.0;
                                        }
                              }
                         else
                              {   V[i][j]=V[i][j];
                                  W[i][j-1] = 0.0;
                              }
              }
          for (i=2;i<=p;i++)
              {   R= R+M[i][g];
              }
          for (j=1;j<=g;j++)
              {   Y=Y+M[1][j];
              }
          for (i=2;i<=p;i++)
              {   X= X+M[i][(2*g+1)];
              }
          for (i=1;i<p;i++)
              {   Z=Z+V[i][g];
              }

          MAKE=R+Y+Z+X;                  // equation (4.4d)
// exclusion of transfer times from the proessing times to display the proces
sing times only
          uiscis2();
          data();
          mntank();
          mutank();
          return 0;
     }

// function to display waiting time inside temporary storage for MIS
```

```
    float mutank()
    {  for (i=1;i<p;i++)
        {   j=g-1;
            if (WW[i][j] == 1)
        //      myfile<<"W" << i << "," << j << "= pass"<<"    ";
            ;   else
        //      myfile<<"W" << i << "," << j << "=" <<W[i][j]<<"    ";
    ;   }
        //myfile<<endl<<"-----------------------------------------------------
--------------------------"<<endl;
        return 0;
    }

// function to display holding time inside stage for MIS

    float mntank()
        {   for (i=1;i<p;i++)
            {   for (j=1;j<g-1;j++)
                //myfile<<"I" << i << "," << j << "=" <<I[i][j]<<"    ";
        ;   }
        //myfile<<endl<<"-----------------------------------------------------
--------------------------"<<endl;
        return 0;
    }
```

APPENDIX F

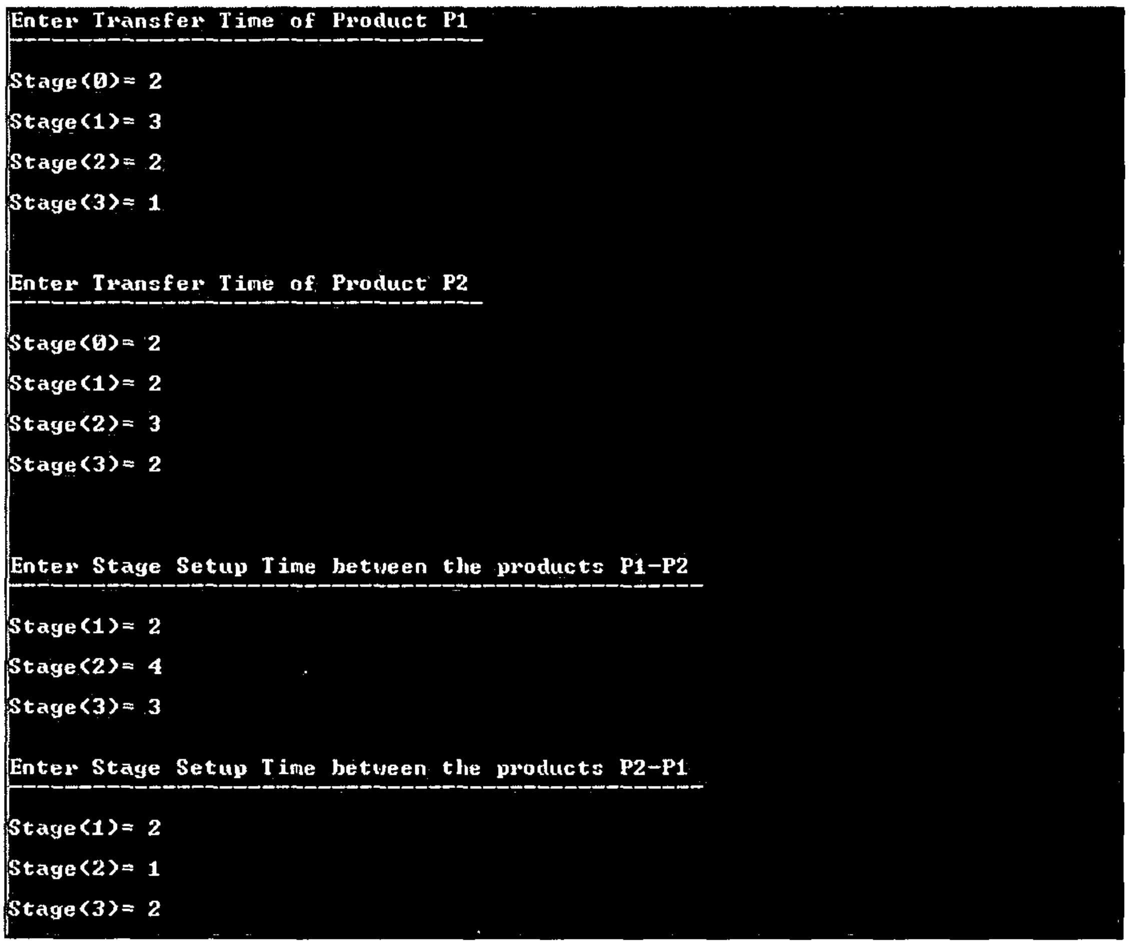
C++ Programming Code Screen Output

Screen Output#1



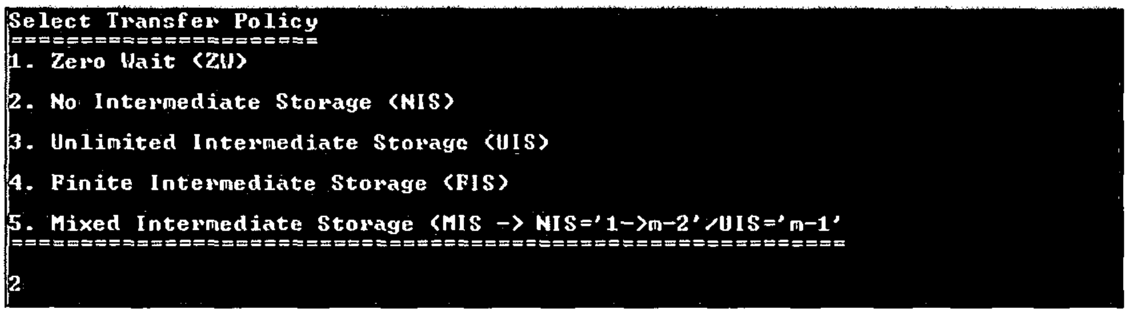
```
*****************************************************************
* Scheduling Multiproduct Batch Process on Makespan Criteria *
*                                                             *
* Compiler: Microsoft Visual C++ ver 6.0                      *
*                                                             *
* Platform: Pentium IV 2.80GHz, Windows XP                    *
*                                                             *
* Programmer: Amir Shafeeq                                    *
*                                                             *
* Date: March 2008                                            *
*****************************************************************


Enter number of Products (Minimum = 2 , Maximum = 100)
2
Enter number of stages   (Minimum = 2 , Maximum = 100)
3

Enter Processing Time of Product P1
------------------------------------
Stage(1)= 2
Stage(2)= 4
Stage(3)= 5


Enter Processing Time of Product P2
------------------------------------
Stage(1)= 6
Stage(2)= 3
Stage(3)= 5

Do you wish to input the transfer and setup time data ? (y/n)
y
```

Screen Output#2



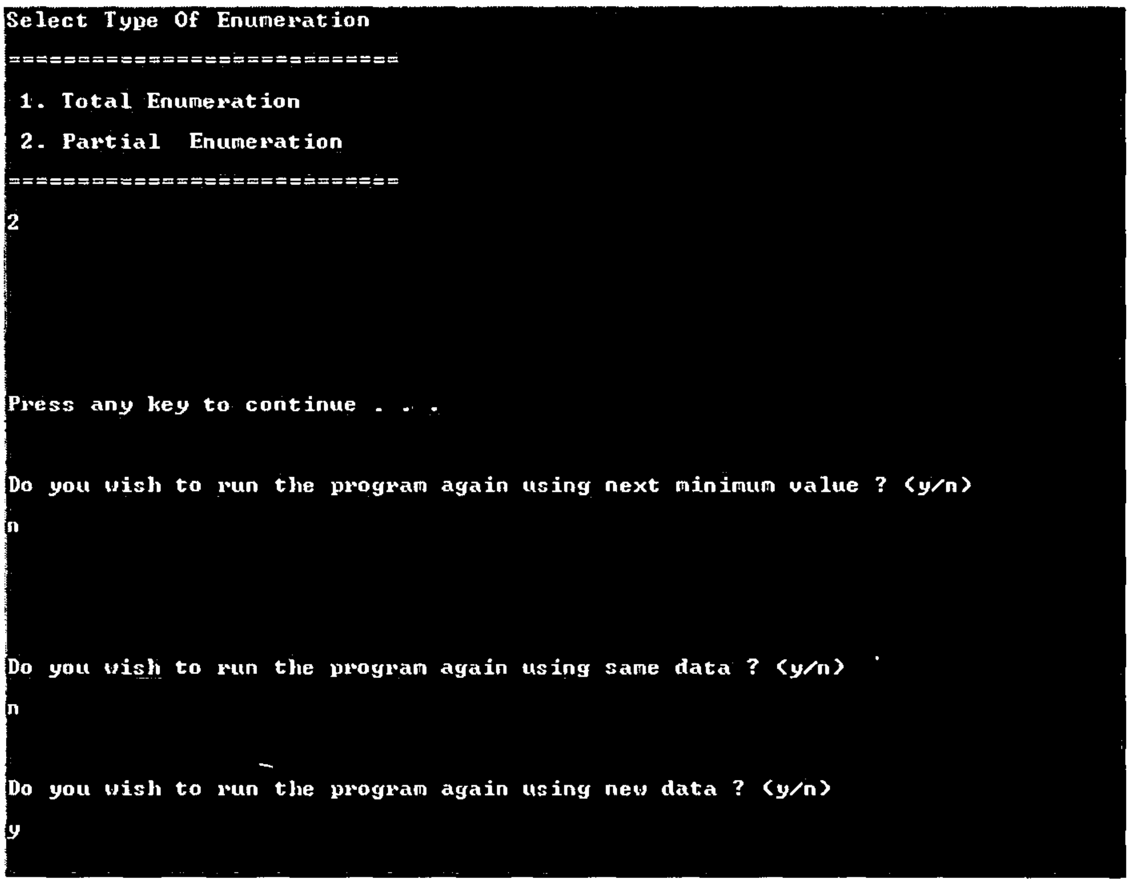
```
Enter Transfer Time of Product P1
---------------------------------

Stage(0)= 2

Stage(1)= 3

Stage(2)= 2

Stage(3)= 1


Enter Transfer Time of Product P2
---------------------------------

Stage(0)= 2

Stage(1)= 2

Stage(2)= 3

Stage(3)= 2



Enter Stage Setup Time between the products P1-P2
-------------------------------------------------

Stage(1)= 2

Stage(2)= 4

Stage(3)= 3

Enter Stage Setup Time between the products P2-P1
-------------------------------------------------

Stage(1)= 2

Stage(2)= 1

Stage(3)= 2
```

Screen Output#3



```
Select Transfer Policy
======================
1. Zero Wait (ZW)

2. No Intermediate Storage (NIS)

3. Unlimited Intermediate Storage (UIS)

4. Finite Intermediate Storage (FIS)

5. Mixed Intermediate Storage (MIS -> NIS='1->m-2'/UIS='m-1'
============================================================

2
```

Screen Output#4



```
Select Type Of Enumeration
==========================
 1. Total Enumeration
 2. Partial  Enumeration
==========================
2




Press any key to continue . . .

Do you wish to run the program again using next minimum value ? (y/n)
n



Do you wish to run the program again using same data ? (y/n)
n

Do you wish to run the program again using new data ? (y/n)
y
```