Optimal Scheduling of an Integrated Batch and Continuous Process System

R.D.M. MacRosty

Submitted in fulfillment of the requirements for the degree of

Master of Science in Engineering

Department of Chemical Engineering
University of Cape Town
Cape Town, South Africa.

Supervisor: A/Prof. C.L.E. Swartz

September 2000

Synopsis

An industrial problem has been posed involving the determination of the optimal scheduling and sequencing of a batch process system. The plant incorporates both batch and continuous units and is further complicated by the existence of a nonlinear relationship between the processing rate and the lot size in the continuous plant. The primary goal of this project is to investigate the application of a rigorous mathematical approach to determine the optimal operating policy of a complex processing system.

In broad terms, scheduling involves the allocation of a limited number of resources to ensure the completion of a set of tasks, in an optimal way. The problem is formulated as a mathematical model through the use of mixed-integer programming. The difficulty associated with this type of work is in describing the plant using constraints to develop a mathematical model which both accurately reflects the plant and renders the problem solvable. The model must incorporate the connectivity of the plant and prevent resource-task allocation conflicts.

A survey of the pertinent literature has been conducted and is reviewed in Chapter 2. Here the principles of batch scheduling are addressed. The focus of the literature review is on the formulation of such problems into a mathematical representation. A description of two fundamentally different approaches for the formulation of short-term scheduling problems is presented. Other aspects of the review include the commercially available software used in the solution of these problems, campaign planning and on-line scheduling.

The work in this thesis initially focuses on the scheduling of a simplified version of the industrial plant. The simplified version consists exclusively of batch processing

units. The motivation for this was to develop a formulation which incorporated the unique characteristics of the full problem but was computationally easier to solve. A number of scenarios were conducted which show both the flexibility of the formulation and the ability of the formulation to reschedule tasks when faced with different operating conditions. In the discussion of these scenarios, issues such as the computational efficiency and the implications of the results are addressed.

The next step was to add a continuous plant upstream to the simplified batch process. The continuous plant exhibits a nonlinear relationship between the processing rate and the efficiency of operation. An approximation of the nonlinearity is proposed and further scenario studies are carried out. Finally, the full problem is tackled but due to the great computational expense required to solve the problem an alternative method is proposed. This method is based on a series of random-type scenarios, which serve as an alternative as well as benchmark to the solutions obtained via rigorous optimization.

Acknowledgements

Firstly, I would like to extend my gratitude to A/Prof. Chris Swartz, for the time and effort he put into the supervision of this project. The continuous support and the wealth of ideas that he contributed to this project proved to be invaluable. I consider myself extremely fortunate to have had such an excellent supervisor, in terms of the rapport he holds with his students and the expertise and effort he puts into each project.

My sincerest gratitude also goes to my wife, Sarah, who has provided me with both motivation and support throughout the duration of this project. Furthermore, I would like to thank her for being so understanding and taking care of the many things that would have otherwise been neglected, particularly during the write-up of this thesis.

I would also like to thank my parents for their encouragement and support. The interest and seriousness with which you have approached my studies is really appreciated. In particular, I would like to thank my father for the discussions and his valiant attempts to understand the details of my project and make suggestions thereon.

A special thanks goes to Gunter Metzner, who organized funding and proposed this project and without whom this project would not have been possible. The industrial application and the pilot plant visits certainly added an interesting dimension to the project.

Next, I would like to thank my fellow students in the Control Group, with whom I was fortunate enough to share an office. The value of the countless conversations

that were held on the balcony I am certain cannot be quantified. Unless one counts the cartons of cigarettes and copious amounts of coffee that were consumed during these sessions. Nonetheless, I would like to thank David Seaman, Rhoda Baker, Kevin Dunn, Ben Knights and Warwick Duncan for both their company and friendship. In particular, I would like to thank David for his late-night companionship in the office and his sharp wit, which served as a great source of amusement.

I would also like to thank Mhairi Kelly and Patrick Smith for their interest, support and friendship. Thanks also to Andrew Prince, Rowan Nairn and Ian McWilliam for their friendship, the 'priceless' emails and the great games of golf, which allowed me to take my mind off of 'things'.

Finally, I would like to thank Craig Balfour, who took care of the technical aspects associated with the software and network administration.

Contents

List of Tables

List of Figures

Nomenclature

Indices

i	tasks
j	units
k	sources
n	event points
s	states
x_i	tasks on continuous plant

Sets

I	tasks
I_j	tasks performed by unit j
J	units
J_i	units suitable to perform task i
K	total number of unique sources
N	total number of event points in horizon
S	set of all involved states

Parameters

C_s	maximum storage capacity for state s
H	time horizon
p_i	processing time of task i
R_{st}	market requirement for state s at time t
r_s	market requirement for state s at the end of time horizon
ST_s^{max}	maximum storage capacity for state s
$V_{ij}^{min/max}$	capacity limitations for task i on unit j
α_{ij}	constant term of processing time of task i at unit j
β_{ij}	variable term of processing time of task i at unit j
$\rho_{is}^{p/c}$	proportion of state s produced/consumed in task i

Variables

B_{ijn}	amount of material undertaking task i in unit j at event point n
B_{ijt}	amount of material undertaking task i in unit j at time t
C_{ij}	completion time of product in thei^{th} position of the sequence processed on unit j
D_{st}	amount of state removed from the system at time t
d_{sn}	amount of state removed from the system at event point n
S_{st}	amount of state s at time t
ST_{sn}	amount of state s at event point n
T^s_{ijn}	time that task i starts on unit j at event point n
T^f_{ijn}	time that task i finishes on unit j that started at event point n
W_{ijt}	binary variable corresponding to the processing of task i on unit j at time period t
wv_{in}	binary variable assigning beginning of task i at event point n
yv_{jn}	binary variable assigning utilization of unit j at event point n
X_{ij}	binary variable corresponding to the processing of task i on unit j
X_{mn}	binary variable corresponding to the set of processing rates, m, used in approximating the operations of the continuous plant at event point, n

Chapter 1

Introduction

1.1 Motivation and Goals

The low volume production of high value products favours batch production. There has been renewed interest in batch operations due to the manufacturing flexibility which these type of operations allow in the production of a large number of different products. In scheduling these processes, it is necessary to allocate limited resources and equipment in an optimal way in order to satisfy production requirements. The scheduling problem has with some degree of success been translated and solved as a mixed-integer linear programming problem. However, these problems can be exceedingly difficult to solve, requiring great computational effort to obtain the optimal solution.

The focus of this thesis is based on an industrial problem which has been posed. For confidentiality reasons, the specifics of the plant have been altered or omitted entirely from this thesis. The plant is a refining process, where the only economic benefit that may be realized is the reduction in the number of personnel hours. Thus the industrial objective of the project is to reduce the number of weekly shifts without sacrificing the production requirements of the plant.

The primary goal of this project is to investigate the application of a rigorous mathematical approach to determine the optimal operating policy of a complex

processing system. This involves the development of a mathematical formulation as a mixed-integer linear programming problem. Further objectives were to investigate factors which influence the computation time, determine the impact of variables on the objective function and also to illustrate the flexibility of the formulation.

1.2 Thesis Overview

Chapter 2

In Chapter 2 a review of work done in the field of scheduling is presented. This encompasses the broad principles of batch scheduling, introducing the reader to both the intricacies and jargon associated with this subject. An overview of mixed-integer programming problems is included in order to make discussions in later chapters easier to follow. A substantial proportion of this chapter is dedicated to the description of the major formulations available in the literature. These formulations are based on the translation of scheduling problems into mathematical models. Following this a brief overview of other aspects associated with scheduling is presented; this includes campaign planning and online scheduling. Finally, a discussion of the available software for the solution of scheduling problems is given. The review is concluded with a short summary, highlighting the important points made in this chapter.

Chapter 3

The focus of Chapter 3 was to apply the formulation to a simplified problem which still exhibited the fundamental characteristics of the full problem. More specifically, the work involved in this chapter focuses on the development of further constraints required to describe the system as well as methods to reduce the computational time required in solving the problem. A number of scenarios were initially conducted using a spreadsheet package; this was done to ensure that the problem was nontrivial. Following this, a number of scenarios were completed solving the problem as mathematical programming problem. These scenarios are aimed at illustrating the implementation of the formulation and also its application to handle possible operational events.

Interesting outcomes which are revealed in the different scenarios are also discussed here.

Chapter 4

This chapter deals with the inclusion of a continuous process which is added upstream. A difficulty arises from the fact that the plant operates in a non-linear fashion as a consequence of its efficiency being related to the processing rate. In order to linearize this nonlinearity a number of approximations can be made. Two different approximations are presented, which are equivalent in terms of accuracy but differ substantially with respect the time taken to solve to completion. The presentation of these approximations is followed by a scenario study.

Chapter 5

In this chapter the formulation is applied to the complete industrial problem. Due to the excessive computational effort required to solve this problem an alternative method is proposed. This involves an approximation method which reduces the time of solution, but does not guarantee optimality. This method relies on the use of a random number generator to determine the sequencing. The random simulation has a dual purpose as it is used to validate the use of optimal scheduling in terms of the quality of the results that may be obtained as well as providing an alternative.

Chapter 2

Review of Batch Scheduling

The first section of this chapter starts by giving a broad introduction into the principles of batch process operations. Following this is a discussion of mixed-integer linear programming. The next section focuses on short-term scheduling and details different methods which are used to formulate and solve these types of problems. A brief analysis of some of the available software that is used to solve these types of problems is also given in this section. Campaign planning and on-line scheduling forms the basis of the next sections.

2.1 Overview of Batch Processes and Scheduling

In this section the principles of sequencing and scheduling of batch process systems is reviewed. This provides a broad overview of the approaches used in batch scheduling that are predominantly followed in the literature. The focus of the review is on the formulation of such problems into a mathematical representation.

Multistage batch plants fall into one of two categories, namely multiproduct or multipurpose plants. In the former the products follow exactly the same sequence. While in the latter, different products may follow different paths through the plant. Figure 2.1 is an example which illustrates the difference between the two flowshops.

In the multiproduct flowshop, product A and product B follow the same path through the plant. In the multipurpose flowshop, product A is processed by a different set of tasks to that of B. Subsequently, these products follow different processing routes through the plant.

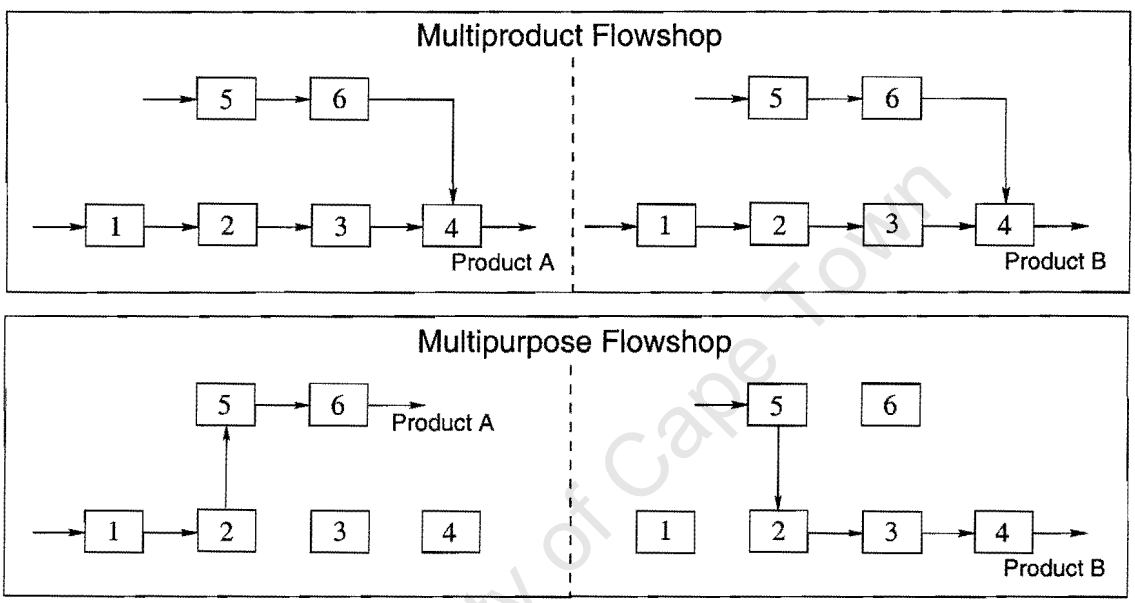


Figure 2.1: Schematic showing the difference between multiproduct and multipurpose flowshops

A preemptive schedule is a schedule where the processing of a product may be interrupted in order to allow the processing of a different product on that unit. The use of a preemptive schedule depends on the nature of the process and of course the storage that is available. A permutation schedule is one in which no passing is allowed, *i.e.* the order in which products are processed remains consistent for all units in the sequence. Valid assumptions that can generally be made in chemical process industry applications are that preemption is not allowed and a permutation schedule is followed (Ku *et al.*, 1987).

Storage Policies The storage facilities available in a batch system form an integral part of the scheduling problem, since they influence the flexibility of the schedule. The available storage options may be classed as follows.

- Unlimited Intermediate Storage (UIS)
- Finite Intermediate Storage (FIS)
- No Intermediate Storage (NIS)
- Zero Wait (ZW)
- Mixed Intermediate Storage (MIS)

The NIS and ZW flowshops are often confused because there is no intermediate storage for either of these policies. Under NIS, there is no intermediate storage although products may remain idle in the unit, once processing is complete, until the downstream unit becomes available. Under the ZW condition the product must be passed from the current unit to the next unit as soon as it has finished processing. This type of storage is normally used where unstable intermediates are produced and must be processed immediately. MIS applies when more than one type of the above mentioned storage facilities exist within a system. The UIS policy is the least restrictive and thus is most likely to provide the greatest productivity. The ZW flowshop on the other hand is the most restrictive storage policy and tends be the least productive flowshop type. The type of storage policies used has a definite influence on the optimal solution.

2.2 Mixed-Integer Linear Programming

This section gives a fairly detailed account of mixed-integer linear programming (MILP). This inclusion of this section is motivated by the fact that in translating a scheduling problem into a mathematical formulation requires the use of integer variables. Furthermore, a number of computational aspects which are unique to integer-based problems are addressed in this thesis, which require an understanding of MILP problems and the method in which they are solved.

MILP problems are similar to linear programming (LP) problems in that an objective function is either maximized or minimized, subject to a number of linear

constraints. However, MILP includes the use of binary variables in the constraints and/or objective function. In scheduling-type problems the binary variables are used to associate the resources to tasks in order to prevent allocation conflicts, such as units processing multiple batches simultaneously. The general mathematical form of a MILP problem is presented in equation (2.1).

$$\begin{aligned} \min \quad f(x,y) &= c^T x + d^T y \\ \text{subject to :} \quad & Ax + By = b \\ & x \geq 0 \quad \text{where } x \in \mathbb{Z}^{n_1} \\ & y \geq 0 \quad \text{where } y \in \mathbb{R}^{n_2} \end{aligned} \tag{2.1}$$

where f is a linear combination of the integer (x) and continuous (y) variables and is the objective function which is to be minimized. This minimization is subject to a set of constraints which are themselves linear combinations of the integer and continuous variables.

2.2.1 The Branch and Bound Algorithm

There is large variety of commercially available software that can be used in the solution of MILP problems. The most widely used and reliable algorithm is the branch and bound method (Willams, 1993). This will be described for the special case where the integer variables are binary.

The problem is initially solved as a linear programming problem by treating the binary variables as continuous variables with a lower bound of zero and an upper bound of 1. This is known as solving the relaxed problem. If the relaxed problem does not return an integer solution a tree-search is initiated. This involves selecting an integer variable whose value, in the relaxed solution, is fractional. These variables are referred to as candidate variables.

The selected variable forms the node of a branch, resulting in the formation of two further sub-problems. On one branch the binary variable is set to a value of 0 and at the other it is set to a value of 1. For each of the sub-problems a new relaxed solution is determined. These sub-problems result in either an infeasible, fractional or integer solution. In the case where a fractional solution is obtained the process is repeated resulting in the generation of further sub-problems. This is illustrated in Figure 2.2. The first selected variable in this example is x_2, which is set to a value of 1 at node 1 and a value of 0 at node 2. Further branching is shown on nodes 1 and 4, where the selected variables are x_1 and x_3, respectively.

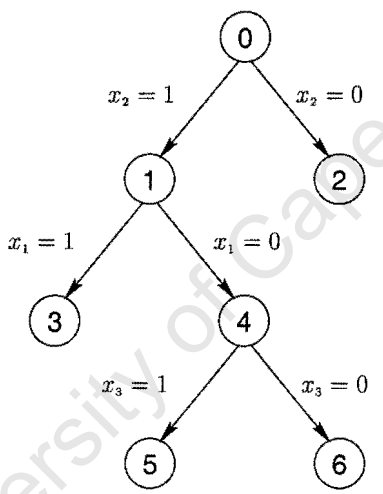


Figure 2.2: Binary search tree with arbitrary branching

The search-tree can have as many as 2^n nodes, where n is the number of binary variables in the problem. This can result in extremely large problems with practically prohibitive computational times. However, there are methods included in the branch and bound algorithm which circumvent the need to carry out a full enumerative search. If an integer solution is found then the objective function value of that sub-problem becomes a bound on the optimal solution. The difference between the objective function for the relaxed problem and the integral solution is known as the 'integrality gap'. As the algorithm progresses down a particular node the variable bounds are tightened (*i.e.* relaxed binary variables are set to 0 or 1) and therefore the solution can only deteriorate. Consequently, if an integer solution has already been found, further searching on the sub-nodes of the integral node will not

produce a better solution. Furthermore, other nodes which have a worse relaxed objective value can be fathomed or pruned from the tree, since no better solution will be found on a sub-branch of that node. Infeasible nodes are also fathomed. The algorithm terminates when all waiting nodes have been explored or once the integrality gap has been reduced to within a specified limit.

There are 3 main alternative approaches for selecting the candidate variable, namely a depth-first search, a breadth-first search or a best bound search, (Floudas, 1995). In a depth-first search the candidate variable will be one of the children nodes of the current node. Backtracking occurs when a node is fathomed due to a integral solution being found or an infeasible solution being obtained. In a breadth-first search all nodes in a given level must be considered before a node is selected from the subsequent level. A best bound search selects the variable which has the smallest lower bound. In general there is a trade-off between the node selection strategies as each have their own advantages and disadvantages.

2.3 Short Term Scheduling

This section begins with a description of how scheduling problems were initially approached, which will be referred to as the classical approach. A major advance in the field of scheduling was made with the work by Kondili *et al.* (1993), whose formulation is based on a discrete-time horizon and is presented in Section 2.3.2. More recently, Ierapetritou and Floudas (1998a) presented a competing formulation which is based on a continuous-time horizon.

Plants which operate under short term production schedules are generally geared towards satisfying individual customer demands within given deadlines. These plants must therefore be flexible since there is no regular production pattern. The time horizons over which these plants operate are generally short and vary from a couple of days to a week. Short term scheduling requires a detailed schedule indicating the resource allocation and the sequence in which products are to be produced. The main goal is to develop an optimal way of utilizing plant resources in order to satisfy

production requirements. A general short term scheduling problem is characterized by Ku *et al.* (1987):

- A set of I products to be produced.
- A set of J processing units.
- A sequence for each product in which operations are to be performed.
- A set of fixed processing times for each product on the processing units.
- A set of fixed transfer times for each product between each set of units.
- Constraints on production order for certain products (precedence constraints).
- A suitable performance criterion to be optimized.
- The nature of intermediate storage between processing units.

The formulation requires the following information:

- Production recipe.
- Available units and their capacity limits.
- Available storage units and their capacity limits.

The objective is to determine the following:

- Optimal sequence of tasks taking place in each unit.
- Amount of material processed at each time in each unit.
- Processing time of each task in each unit.

Objective Function

In terms of the performance criterion a number of different objective functions can be used in order to determine an optimal schedule. The makespan, mean flowtime, the maximum tardiness and economic criterion are the most common objective functions associated with this type of problem. The makespan is the time it takes for the last product to finish processing on the last unit. The mean flowtime is defined as the time required to pass completely through the process, averaged over all the products. Tardiness is defined as the difference between the delivery date of a product and its due date. These first three criteria should be minimized, whereas economic based objective function must be maximized in order to determine the optimal schedule.

2.3.1 Classical Formulation

In the review paper by Ku *et al.* (1987) the general method in which scheduling problems have traditionally been formulated is presented. This type of approach is somewhat dated but can be used for planning simple batch operations where a number of fairly similar products are processed via a small number of tasks, (Kondili *et al.*, 1993). The method presented by Ku *et al.* (1987) is summarized below.

The variables used in the formulation are defined as follows:

- C_{ij} is the completion time of the product in the i^{th} position of the sequence on the j^{th} unit.
- T_{kj} is the processing time of the product k on the j^{th} unit, which is known.
- X_{ki} is a binary variable which is 1 when product k is in position i in the sequence and 0 otherwise.

Figure 2.3 presents a short-term schedule for the processing of 4 products on a simple 3-stage plant. All 4 products follow the same processing sequence, *i.e.* stage 1, stage

2 and then stage 3. In the figure, the duration of each task (T_{kj}) is represented by the length of the corresponding bar, while the completion time (C_{ij}) of each task coincides with the end of the bar. In this particular example there is no intermediate storage. Consequently, products are required to remain in the processing units until the downstream unit becomes available. These idle periods where units are unproductively occupied are depicted as plain white bars in Figure 2.3.

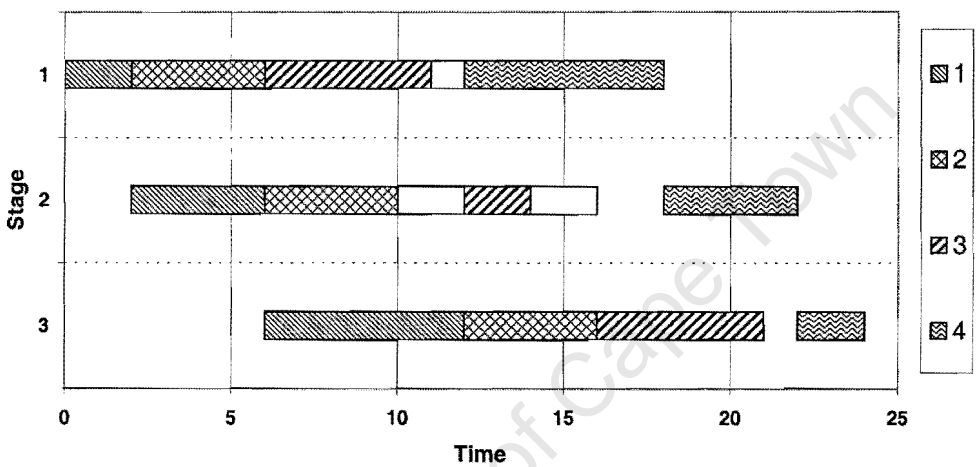


Figure 2.3: Gantt Chart showing the schedule for a simple process plant

Objective Function:

The objective function, Z, to be minimized is the makespan. Assuming there are I products and J units in total then the objective function is given by equation (2.2), which is the completion time of the product that is processed last in the sequence on the last unit.

$$Z = C_{IJ} \tag{2.2}$$

Constraints:

There are a number of constraints which are required in order to describe the system as a MILP problem. These constraints are detailed below.

1. Allocation Constraints

Equation (2.3) ensures that each product is assigned to exactly one position in the product sequence.

$$\sum_{i=1}^{I} X_{ki} = 1 \qquad \forall\, k \in I \tag{2.3}$$

Equation (2.4) ensures that each position is assigned to only one product.

$$\sum_{k=1}^{I} X_{ki} = 1 \qquad \forall\, i \in I \tag{2.4}$$

where I is the number of available positions in the sequence.

2. Recurrence Relations

The following set of constraints are termed recurrence relations and are used to calculate the completion times for the production sequence. It should be noted that these relations assume that there is unlimited storage between units.

$$C_{ij} = \underbrace{\max\left[C_{i-1,j}, C_{i,j-1}\right]}_{\text{min start time}} + T_{kij} \tag{2.5}$$

where the braced term is the minimum starting time of the product in the i^{th} position of the sequence on the j^{th} unit and T_{kij} refers the processing time of product k which is processed in this position on unit j.

As initial conditions we have the two following constraints.

$$\begin{aligned} C_{i0} &= 0 \qquad && \forall\, i \in I \\ C_{0j} &= 0 \qquad && \forall\, j \in J \end{aligned} \tag{2.6}$$

To express these criteria as constraints in the MILP formulation, the following decomposition is necessary, due to the presence of the discontinuous max function.

$$C_{ij} \geq C_{i-1,j} + \sum_{k=1}^{I} T_{kj} X_{ki} \qquad \forall\, i \in I \tag{2.7}$$

$$C_{ij} \geq C_{i,j-1} + \sum_{k=1}^{I} T_{kj} X_{ki} \qquad \forall\, i \in I \tag{2.8}$$

Equation (2.7) states that the completion time of any product must be greater or equal to the sum of the completion time of the previous job processed on a unit and the processing time of the current product on the unit. The summation term in equation (2.7) reduces to the processing time of the product k which is processed i^{th} in the sequence on the j^{th} unit, since the binary variable X_{ki} will be zero for all products other than product k at the i^{th} position in the sequence.

Equation (2.8) ensures that the completion time of product i on unit j is greater or equal to the processing time of that product on the previous unit.

The initial condition is as follows,

$$C_{ij} = 0 \qquad for\ i \geq 0 \tag{2.9}$$

An alternative set of recurrence relations was proposed by Mah (1990), which incorporates the NIS policy.

$$C_{ij} = \underbrace{\max\left[C_{i,j-1}, C_{i',j+1} - T_{k_{i'},j+1}\right]}_{\text{min start time}} + T_{k_i j} \tag{2.10}$$

where the braced term is the minimum starting time of the product in the i^{th} position of the sequence on the j^{th} unit and $T_{k_i j}$ refers the processing time of product k which is processed in this position on unit j. The index i' refers to the previous job being processed on the downstream unit, $j+1$.

Here, the completion time (C_{ij}) of job i is dependent on the completion time of the same job on the previous unit ($C_{i,j-1}$) as well as start time of the job on the downstream unit ($C_{i',j+1} - T_{k_{i'},j+1}$) which is the condition for NIS. This implies that the product cannot be removed from a particular unit until the downstream unit has finished processing.

The initial conditions are as follows,

$$\begin{aligned} C_{i0} &= 0 && \forall\, i \in I \\ C_{0j} - T_{0j} &= 0 && \forall\, j \in J \\ C_{iJ} &= C_{i,J+1} - T_{k_i,J+1} && \forall\, i \in I,\ j \in J \end{aligned}$$

where the last condition is included to ensure the validity of equation (2.10) on the last unit.

2.3.2 Discrete-Time Formulation

The approach presented by Ku *et al.* (1987) is useful for determining the scheduling in a simple batch system. However, its usefulness is limited for the application to more complex flowshops which include intricacies such as the mixing and splitting of streams. A number of attempts for producing algorithms for both the sequencing and scheduling have been published. However, many of these attempts are limited by various assumptions that are made in the formulation of the problem.

In work done by Egli and Rippin (1986) and Rich and Prokopakis (1986) the simultaneous batch sequencing and scheduling problem was approached by imposing certain restrictions on the problem. With the incorporation of these restrictions the generality of the problem is lost and the formulation is only valid for the selection of plants which abide by these restrictions. In particular, these formulations are based on plants in which no mixing or splitting of batches is allowed and storage policies are limited. Wellons and Reklaitis (1991) formulate the problem by allocating equipment to tasks by creating groups of units. The groups for a task then operate out of phase, while the items within a task operate in phase with each other. Each item of equipment may only be used for one task in each phase.

Kondili *et al.* (1993) made a large contribution to the field of optimal scheduling where they present a general formulation that considers the scheduling, and sequencing problem. In this formulation batches may be split or merged as desired and all types of storage policies may be encompassed into the formulation. Another important point in this formulation, is that resource assignment is not pre-determined, such as the approach followed by Wellons and Reklaitis (1991); rather it is determined in the solution of the problem.

The problem is formulated through the use of a state-task network (STN) representation, (Kondili *et al.*, 1993), which forms the framework of the formulation. The STN can be manipulated for the treatment of complex flows, such as the cross-linking of product lines and recycling of materials. This representation is particularly useful in that it eliminates the ambiguities that arise when representing a system of recipe networks in a typical flowsheet. A description of the STN follows.

2.3.2.1 State Task Network Representation

The STN framework was first presented by Kondili *et al.* (1993) and has subsequently become a standard representation in the field of optimal scheduling, forming the basis of the more recent work in this field. A STN representation is similar to a standard process flowsheet representation, but is intended to describe the process rather than the actual physical plant. In fact, the process units and their connectivity are not explicitly shown in the STN representation. State nodes (depicted by

circles) represent the material at the various stages of processing i.e. feeds, stable intermediates and products. In many respects the states may be viewed as storage stations. The type of storage which these states represent is determined in the formulation of the problem. Task nodes (depicted as rectangles) represent operations which transform material from one or more input states into one or more output states.

Figure 2.4 is a simple example of how a process flow diagram (PFD) can be converted into a STN diagram. As far as the production recipe is concerned, the PFD is ambiguous in that it is uncertain whether two distinct products are produced or a single product which is just split between units 1 and 2. Figure 2.4 indicates how the STN framework can be used to remove these ambiguities in order to describe the process recipe. In the first STN representation, STN(a), task 1 produces two distinct products which are then processed separately in units 2 and 3. In the second representation, STN(b), task 1 only produces a single product, which is then split and processed in units 2 and 3.

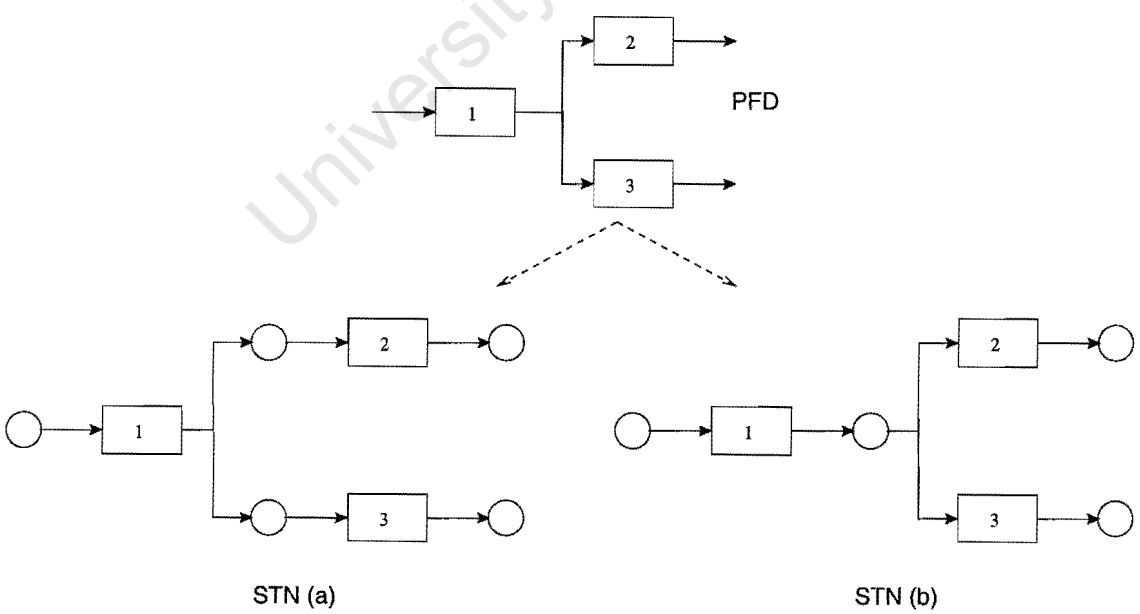


Figure 2.4: Conversion of a process flow diagram to a STN diagram

In translating a plant into the STN framework the following rules must be adhered:

- A task has the same number of input (output) states as different types of input (output) material.
- Streams entering the same state must be of the same quality. If mixing of different streams occurs, this must be included as a separate task *i.e.* mixing cannot occur in a storage unit.

Kondili *et al.* (1993) suggest modifications to the STN framework in order to handle various complexities that may arise. Modifications that are relevant to the work carried out in this thesis include the handling of the multipurpose storage facilities and batch-size dependent processing times.

Multipurpose Storage In the standard STN framework it is assumed that each task has its own dedicated storage. However, in many instances there are only a limited number of storage units which must be shared by groups of tasks. Kondili *et al.* (1993) propose the implementation of a storage task, which receives material from a state (such as a storage unit in reality) and produces the same amount of material during the next interval. The storage tasks are treated as any other task and their use is thus associated with a binary variable. This allows the introduction of storage capacity constraints which are product specific or allow the simultaneous storage of only certain products. An example of a modification to the STN which encompasses storage tasks is shown in the following diagram (*viz.* Figure 2.5) where *ST* represents the storage task.

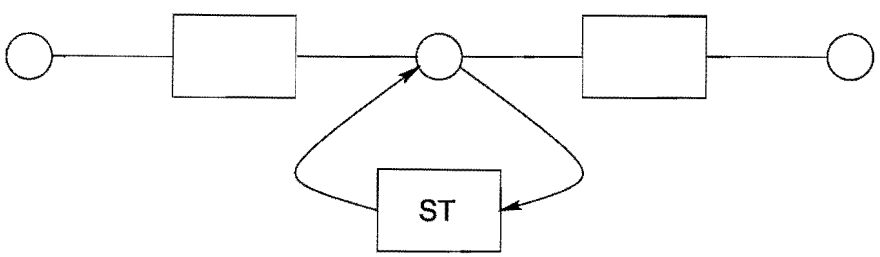


Figure 2.5: Modification of the STN to encompass multipurpose storage

Batchsize Dependent Processing Times The formulation presented by Kondili *et al.* (1993) is intended for cases where the processing times are fixed and are known *a priori*. In cases where the processing time is dependent on the size of the batches, the use of a piecewise approximation function is suggested. Thus the interval between the maximum and the minimum capacity of the unit is divided into smaller sub-intervals. A processing time is then associated with each of these sub-intervals. This can be encompassed into the STN framework by representing each sub-interval as a task with the respective processing rate and capacity limitations, which coincide with the upper and lower limits of that sub-interval. An example of this modification is show in Figure 2.6. The batch-size dependent task (task 1) is discretized into 3 tasks, namely tasks 1a, 1b and 1c.

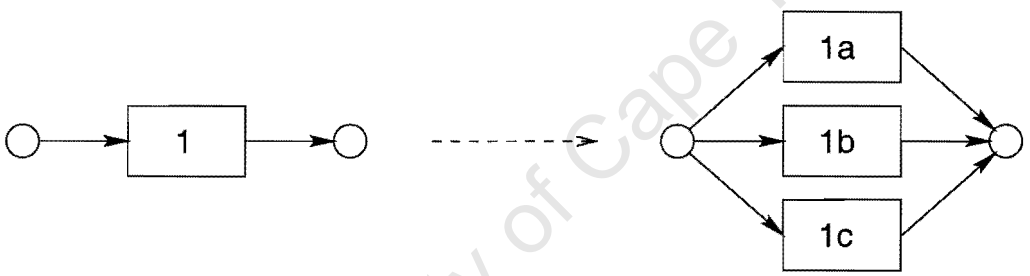


Figure 2.6: Modification of the STN for batch-size dependent processing times

2.3.2.2 Mathematical Formulation

The formulation is based on a discretization of the time horizon. The horizon is thus divided up into a number of time intervals of equal and fixed duration. The size of the intervals is usually the highest common factor of the processing times involved. All events, such as the beginning and end of tasks, must coincide with the interval boundaries, thus ensuring that resource limitations are satisfied at all times. The basis of the formulation is to relate the utilization of resources over the time horizon, while taking into account all the constraints. In order to determine the optimal schedules of batch operation it is assumed that the material received by a task from its feed states is fixed and known, as is the material it produces to the

output states. The processing time for each product is known and is assumed to be independent of size. However, different products may have different processing times on a particular unit.

Constraints

The following section details the set of constraints must be satisfied.

1. Allocation Constraints

To ensure that a piece of equipment processes only one task at a time the following mathematical constraint is implemented in the formulation.

$$\sum_{i \in I_j} W_{ijt} \leq 1 \tag{2.11}$$

where W_{ijt} is a binary variable which is equal to one when task i takes place on unit j beginning at time interval t, otherwise it is zero. I_j is the set of tasks which can be performed by unit j.

To prevent a preemptive schedule (*i.e.* a schedule where the processing of a task is interrupted in order to process another task) from occurring, equation (2.12) is included in the formulation.

$$\sum_{i' \in I_j} \sum_{t'=t}^{t+p_i-1} W_{i'jt'} - 1 \leq M\,(1 - W_{ijt}) \qquad \forall j, t,\ i \in I_j \tag{2.12}$$

where M is a sufficiently large positive integer. This constraint ensures that if $W_{ijt} = 1$, then for any task $i' \in I_j$ and $t' = t+1, ...t+p_i-1$, then $W_{i'jt'} = 0$. Thus for the entire duration of task i on unit j, no other task may interrupt processing. p_i is the processing time of task i.

2. Capacity Limitations for Units and Storage Facilities.

The capacity constraints for a task i being processed on unit j are as follows.

$$W_{ijt}V_{ij}^{\min} \leq B_{ijt} \leq W_{ijt}V_{ij}^{\max} \qquad \forall i, t,\ j \in J_i \tag{2.13}$$

where B_{ijt} is the amount of material undergoing task i on unit j at the beginning at time period t. J_i is the set of units capable of performing task i.

The inclusion of the binary variable forces the batch size to zero if $W_{ijt} = 0$, thereby ensuring that no material is processed if the task does not begin at time t. However, if the task does begin at time t (*i.e.* $W_{ijt} = 1$), then the quantity of material must be less than $V_{ij}^{\max}$.

The storage capacity is constrained in the following manner.

$$0 \leq S_{st} \leq C_s \qquad \forall s, t \tag{2.14}$$

where S_{st} is the amount of material stored in state s at the beginning of time period t and C_s is the maximum storage capacity of storage unit s.

From the above relationship it is possible to see how all types of flowshops (UIS, FIS, *etc.*) can be easily integrated into the problem formulation.

3. Material Balance

The connectivity of the plant and the flow of the material is controlled through the inclusion of a material balance, *viz.* equation (2.15).

$$S_{st} = S_{s,t-1} + \sum_{i \in \overline{T}_s} \rho_{is}^{p} \sum_{j \in J_i} B_{i,j,t-p_{is}} - \sum_{i \in T_s} \rho_{is}^{c} \sum_{j \in J_i} B_{i,j,t-p_{is}} - D_{st} + R_{st} \qquad \forall s,t \qquad (2.15)$$

where ρ_{is}^{p} and ρ_{is}^{c} are the proportion of input and output of task i from state s, respectively. $\overline{T}_s$ is the set of tasks producing material in state s, whereas T_s is the set of tasks requiring material from state s. p_{is} is the processing time of task i that reports to state s. D_{st} is the amount of product delivered to the market and hence removed from the system. R_{st} is the amount of raw material added to the storage unit during the time horizon.

The above equation is in the form of a typical mass balance. The material in storage unit s, at time t is equal to the amount of material that was present in the previous time interval, minus the amount of material moved to downstream processing units, plus the amount of material added from the upstream units. The amount of material removed to the market and the amount of raw material are subtracted and added, respectively.

The allocation, capacity and mass balance constraints form the basis of the mathematical model. However, Kondili *et al.* (1993) formulate a number of other operational constraints to handle situations such as unavailability of equipment due to maintenance, limited utilities and manpower and the cleaning of equipment items.

Objective Function

As is common with these types of problems the objective function is based on either an economic or system performance measure. The criterion used by Kondili *et al.* (1993) is the maximization of profit which is expressed in equation (2.16). However,

any of aforementioned performance criteria may be used in this formulation.

$$\begin{aligned}\text{Profit} = &[\text{value of products}] - [\text{cost of feedstock}] \\ &- [\text{cost of storage}] - [\text{cost of utilities}]\end{aligned} \tag{2.16}$$

2.3.2.3 Computational Issues

A characteristic of MILP problems is that they often require immense computational effort to solve. The benefit of using a discretized time formulation is that the level of discretization can be manipulated in order to reduce the ultimate size of the problem. However, there is a trade off between computational efficiency and the level of accuracy. Problems which are based on a time step that is too large to accurately reflect the plant in the formulation, may find an optimal solution which is in fact a sub-optimal solution for the real case.

In the work of Shah *et al.* (1993b), a number computational improvements which do not affect the optimality of the solution are presented. These improvements address issues such as the reduction of the integrality gap through the reformulation of some of the constraints and *a posteriori* analysis of the problem. By tightening the relaxation gap the search space for integer solutions is reduced. Further reductions were obtained by using a modified branch and bound procedure which exploits a number of the characteristics of scheduling problems, resulting in much reduced computational times. Yee and Shah (1998) propose improvements in the case where change-over activities are responsible for increasing the integrality gap by the inclusion of additional constraints, ensuring that a minimum number of change-over tasks occurs.

2.3.3 Continuous-Time Formulation

Formulations based on a continuous representation of time are reported to exhibit large integrality gaps compared to their discrete-time counterparts, (Shah *et al.*, 1993b; Schilling and Pantelides, 1996; Yee and Shah, 1998). Large integrality gaps

tend to worsen the computational efficiency of these problems. However, there are a number of advantages of using a continuous-time representation. For example, discrete-time formulations may require a much greater number of discrete variables in order to obtain the same degree of accuracy.

There have been a number of continuous-time formulations published, which exhibit the same degree of generality as the discrete-time formulation proposed by Shah *et al.* (1993a). Zhang and Sargent (1994) and Schilling and Pantelides (1996) presented formulations based on a continuous-time representation. Both formulations are based on the resource-task network (RTN) framework. Pantelides (1994) introduced the idea of the RTN, which is a modification of the STN representation.

In the RTN framework there is no distinction between the units or any other available resources, such as raw materials, utilities and so forth. All resources are thus treated the same and may be consumed or produced at any time. Therefore units are treated as being consumed by the initiation of a task and then produced at the termination of the task. In cases where a utility such as steam is involved, it may be the case that only a portion of the amount available is required for a particular task. To handle this situation a variable is introduced, which accounts for the 'extent' that each resource is used. This above discussion provides a brief background into the RTN framework, but for a more detailed discussion the reader is referred to the paper by Pantelides (1994).

The formulations by Zhang and Sargent (1994) and Schilling and Pantelides (1996) both resulted in problems requiring substantial effort to solve when compared to the discrete-time formulation of Shah *et al.* (1993a). An alternative formulation was proposed by Mockus and Reklaitis (1997) which is based on the STN framework. This formulation was based on using a non-uniform discretization of the time horizon. However, the formulation yields a mixed-integer nonlinear programming problem which is reduced to a mixed-integer bilinear programming problem. Unfortunately the bilinear problem is non-convex and therefore global optimality is not guaranteed.

More recently, Ierapetritou and Floudas (1998a) proposed a continuous-time formulation which reduces much of the computational burden experienced with other

continuous-time formulations. This formulation is the most recent of the continuous-time formulations and has also been implemented with the most success. A detailed account of the formulation is thus presented.

2.3.3.1 Mathematical Formulation

This formulation is based on the STN framework and makes use of 'event points' which form a parallel co-ordinate to prevent resource-task allocation conflicts. These event points are independent of time and correspond to either the initiation of a task or unit utilization. The number of event points required is determined by an iterative procedure, which simply involves increasing the number of points until the objective function no longer improves, indicating that there are sufficient event points to handle the resource-task conflicts. Another important feature of this formulation is the use of separate binary variables to represent task events and unit events. The benefit of decoupling the task and unit events is that the number of binary variables is decreased and nonlinearities associated with scheduling through parallel units are avoided (Ierapetritou and Floudas, 1998a).

For the purpose of the presentation of this formulation, we distinguish between the sets I and I_j. I_j is a subset I, where I_j is the set of suitable tasks that can be processed on unit j. Similarly, J_i is the subset of units, J, suitable for task i.

1. Allocation Constraints

$$\sum_{i \in I_j} wv_{in} = yv_{jn} \qquad \forall\, j \in J,\ n \in N \qquad (2.17)$$

wv and yv are the binary variables associated with tasks and units, respectively. wv_{in} is equal to one if task i was initiated at the beginning of event point n, otherwise it is equal to zero. Similarly, yv_{jn} is equal to one if unit j began operating during event point n, otherwise it is equal to zero. These constraints assign any number of tasks to a specific unit over the scheduling horizon, however, at most only one

task may begin at a specific event point. For example if unit j is utilized ($yv_{jn} = 1$) at event point n, then a single task from the subset of tasks ($i \in I_j$) that may be completed on that unit will begin at that event point. The problem of obtaining a pre-emptive schedule does not arise in this formulation due to the method in which the task timing is determined.

2. Capacity Constraints

$$V_{ij}^{\min} wv_{in} \leq B_{ijn} \leq V_{ij}^{\max} wv_{in} \qquad \forall\, i \in I,\ j \in J_i,\ n \in N \tag{2.18}$$

The variable $B_{i,j,n}$ is the amount of material undertaking task i on unit j at event point n. The above constraints ensure that the amount of material being processed in a unit does not exceed the capacity limitations of that unit. The inclusion of the binary variables forces the amount of material being processed to be zero in the case where the task is not being performed at that particular event point.

3. Storage Constraints

$$ST_{sn} \leq ST_s^{\max} \qquad \forall\, s \in S,\ n \in N \tag{2.19}$$

The variable ST_{sn} represents the amount of material stored in state (storage unit) s at event point n. The storage constraints ensure the physical capacity of the storage units is not exceeded at any time during the time horizon.

4. Material Balance

$$ST_{sn} = ST_{s,n-1} + \sum_{i \in I_s} \rho_{si}^{p} \sum_{j \in J_i} B_{ij,n-1} - \sum_{i \in I_s} \rho_{si}^{c} \sum_{j \in J_i} B_{ijn} - d_{sn} \qquad \forall\, s \in S,\ n \in N \tag{2.20}$$

where I_s is defined as the set of tasks which process state s. ρ_{is}^p and ρ_{is}^c are the proportion of input and output of task i from state s, respectively. d_{sn} is the amount of material removed from state s at event point n.

The mass balance constraint can be described as follows. The amount of material in a particular storage unit (state) is equal to the amount of material present at event point $n-1$ and the amount of material processed in the upstream unit during the previous event point, minus the amount of material removed from the system (d_{sn}) and the amount transfered to the downstream unit. The material balance forms the basis of the formulation, in that it takes into account the quantity of material stored in each state and also how much material is processed in each unit at each event point.

5. Demand Constraints

$$\sum_{n \in N} d_{sn} \geq r_s \qquad \forall\, s \in S \tag{2.21}$$

These constraints require the amount of material that leaves the system to be at least that required by the market.

6. Duration Constraints

$$T_{ijn}^f = T_{ijn}^s + \alpha_{ij} w v_{in} + \beta_{ij} B_{ijn} \qquad \forall\, i \in I,\ j \in J_i,\ n \in N \tag{2.22}$$

The variables T_{ijn}^s and T_{ijn}^f represent the starting and finish times of tasks, respectively. The above equation is used to calculate the finishing time of a task, based on the starting time and the processing time of the task. Here, α_{ij} is the processing time and β_{ij} is the variable term of the processing time. The term, $\beta_{ij} B_{ijn}$ is included to account for process variability based on the size of the batch.

7. Sequence Constraints: Same Task in the Same Unit

$$T^s_{ij,n+1} \geq T^f_{ijn} - H\,(2 - wv_{in} - yv_{jn}) \qquad \forall\, i \in I,\ j \in J_i,\ n \in N,\ n \neq N \tag{2.23}$$

H is a parameter and is the time horizon over which the scheduling is done. The above constraint ensures that the starting time of a task at the event point $(n+1)$ is greater or equal to the finishing time of the previous task that was started at event point n. The inclusion of the second term (on the right hand side) relaxes the constraint in the event that task i does not take place and unit j is not used at event point n. If the constraint is relaxed the following two constraints (2.24 and 2.25) ensure the correct timing is maintained.

$$T^s_{ij,n+1} \geq T^s_{ijn} \qquad \forall\, i \in I,\ j \in J_i,\ n \in N,\ n \neq N \tag{2.24}$$

$$T^f_{ij,n+1} \geq T^f_{ijn} \qquad \forall\, i \in I,\ j \in J_i,\ n \in N,\ n \neq N \tag{2.25}$$

8. Sequence Constraints: Different Task in the Same Unit

$$T^s_{ij,n+1} \geq T^f_{i'jn} - H(2 - wv_{i'n} - yv_{jn}) \qquad \forall\, j \in J,\ i \in I_j,\ i' \in I_j,\ i \neq i'\ n \in N \tag{2.26}$$

The form of the above constraint is similar to that of equation (2.23), except that it applies for different tasks performed on the same unit. In this constraint the relationship between two tasks (i and i') at points n and (n+1) is established. If no task takes place on unit j at event point n, then the second term on the right hand side ensures that constraint is trivially satisfied and the starting time is dependent on the other constraints.

9. Sequence Constraints: Different Tasks in Different Units

$$T^s_{ij,n+1} \geq T^f_{i'j'n} - H(2 - wv_{i'n} - yv_{j',n})$$
$$\forall\, j, j' \in J_i,\ i \in I_j,\ i' \in I_{j'},\ i \neq i', n \in N,\ n \neq N \tag{2.27}$$

This constraint is similar to that of equation (2.26), except that this constraint concerns different tasks on different units. This constraint handles the task precedence and is thus only applicable to tasks which occur sequentially. More specifically, the start time of a task must be greater than the completion time of the preceding task on the previous unit in the production sequence of a product.

10. Sequence Constraints: Completion of Previous Tasks

$$T^s_{ij,n+1} \geq \sum_{n' \in N, n' \leq n} \sum_{i' \in I_j} \left(T^f_{i'jn'} - T^s_{i'jn'} \right)$$
$$\forall\, i \in I,\ j \in J_i,\ n \in N,\ n \neq N \tag{2.28}$$

The above constraint ensures that a process can only begin after the completion of all previous tasks on that unit.

11. Time Horizon Constraints

$$T^f_{ijn} \leq H \qquad \forall\, i \in I,\ j \in J_i,\ n \in N$$
$$T^s_{ijn} \leq H \qquad \forall\, i \in I,\ j \in J_i,\ n \in N \tag{2.29}$$

These constraints ensure that the start and finish times occur within the given time horizon, H.

12. Objective Function

The objective function is similarly constructed to that of the discrete-time formulation described in Section 2.3.2 and can be either the minimization of the makespan or the maximization profit.

2.4 Other Aspects of Scheduling

2.4.1 Campaign Planning and Scheduling

An overview of campaign planning is detailed in this section. However, the mathematical formulation of this type problem is beyond the scope of this review.

Campaign planning is suitable for plants operating under stable demand conditions and where the plant may be dedicated to a small subset of its potential products for relatively long periods of time. The planning horizon is divided into a series of time periods or production campaigns. A production line is described as a periodic sequence of batches that operate for an entire campaign period. Production lines of different products that do not have conflicting resource requirements are grouped together to form a campaign. Each campaign is described by a periodic schedule during which the plant resources are dedicated to the production of a single product or a subset of products. A cyclic pattern of operation is established during each campaign, where identical batches are produced in sequence. Campaign planning therefore involves the allocation of the available production time among the various campaigns and the determination of a detailed periodic schedule of each campaign.

Traditionally, this type of problem has been solved by decomposing the problem into two levels (Mauderli and Rippin, 1979). In this formulation it is assumed that the units are multi-purpose and a ZW policy is followed for the transfer between units. On a macro level, the production plan consists of a number of production targets. The solution procedure is to generate a number of campaigns and to discard the inefficient ones. The next step is to screen the campaigns to determine

the dominant ones. The allocation of time to the dominant campaigns is then solved as a linear programming problem. The lower level problem involves the allocation of resources to tasks in order to meet these production targets. Other authors have proposed modifications to this strategy. However, in all this work it is assumed that there is a linear sequence of processing tasks with no intermediate storage between them. These assumptions restrict the solution, in that if a product is produced in a campaign then all its constituent tasks must occur within that campaign. Also, within a specific campaign a unit cannot be used for more than one task. These assumptions can be limiting, particularly for processes involving a number of processing steps.

Shah and Pantelides (1991) take a more general approach and solve the production planning and campaign construction problems simultaneously. The manufacture of each product is decoupled into several stages through the use of intermediate storage. Each stage consists of several tasks, which can then be run independently in campaign mode. The problem is formulated as a MILP taking into consideration both the production planning and the construction of the campaigns.

Traditionally, it has been accepted that the optimal way of operating plants with long planning horizons, is in a periodic mode where the same sequence of tasks are executed repeatedly. Shah *et al.* (1993a) propose that the optimal operating schedule for a plant, from an economic point of view, may be to operate in a non-periodic mode. However, the non-periodic schedule is complex to determine and may result in over complicating the operation, which is impractical. However, in certain instances it may be possible to apply this approach of non-periodic planning.

2.4.2 On-Line Scheduling

In determining the schedule off-line, it is assumed that the plant will operate exactly as it is scheduled. However, this is rarely the case since batch processes inherently exhibit a degree of variability (Cott and Macchietto, 1989). This variability can be due to a number of factors, such as the fluctuation in utility availability, equipment malfunctions, recipe inaccuracies and changes in raw material quality and quantity.

Furthermore, due to the solution time required in determining the optimal schedule it is not viable to re-solve the problem and implement changes in real-time. For this reason the majority of work dealing with on-line scheduling implements reactive-type heuristical procedures to decrease deviations from the optimal schedule determined off-line. The use of off-line scheduling can have the two following effects.

- If the process time for a task is longer than the scheduled one, the time spent by batches waiting for processing units may increase.
- If the process time for a task is shorter than the scheduled one, the idle time of process units may increase.

Karimi and Reklaitis (1985) proposed a method to minimize the effects of using off-line scheduling, by introducing intermediate storage and maintaining a reserve amount of material in storage. However, this is not a viable option in the case where a zero wait policy is followed or in a multiproduct plant, since storage units would then be required for each different intermediate product.

Typically, processing times used in scheduling are averages based on previous runs. Therefore the accuracy of off-line scheduling is questionable. Cott and Macchietto (1989) propose the use of an on-line schedule where the original schedule is dynamically modified as unpredicted deviations between the original schedule and the actual operation become apparent.

The general approach to handling the on-line scheduling problem is based on detecting deviations and then making modifications to the schedule. This is analogous to feedback control theory. Cott and Macchietto (1989) approach the on-line scheduling problem in this way, by altering the start times of future batches. This method requires that the process is monitored and compared to the original schedule to determine if there are any deviations. An algorithm is then applied in order to determine how the starting times of the remaining batches should be modified. For these modifications to be implemented in real-time, the algorithms cannot be overly complex or time consuming to run.

Cott and Macchietto (1989) proposed a number of simple algorithms, which are based on the shifting the schedule by a time depending on the size of the deviation between the actual process and the original schedule. The most successful of these algorithms is referred to as the 'shift-advance' schedule. In this algorithm the value of the deviation is only considered if it is negative, i.e. the process time is shorter than that determined by the schedule. The algorithm then compresses the schedule in an attempt to recover idle times. After experimental work conducted by the authors, they concluded that these algorithms based on simple time shifts lacked sufficient detail to minimize the effects of both the wait times (products) and the idle times (units).

Subsequently Cott and Macchietto (1989) developed an algorithm with a similar theoretical basis to that used in model predictive control theory. The algorithm looks at the current operation and predicts future completion times of all executing batches. From the estimated start times of future batches, the algorithm searches for wait times as well as recoverable idle times. The schedule is then readjusted accordingly. The authors report this algorithm to be an improvement on the simple time shift model, originally proposed. It should be noted that the above algorithms are not appropriate for major process disruptions such as a processing unit going out of service, which would require complete rescheduling.

Kanakamedala *et al.* (1994) considered this problem, where rescheduling was considered by taking into account the possibility of unit reassignments. The schedule modification is based a decision tree subject to a beam search which aims to minimize deviation from the original schedule.

Ultimately, one would rather re-determine the schedule in the event of deviations. However, due to the long solution times of these problems and the state of current technology it is not possible to re-determine and implement the results in real-time.

2.5 Software for Solving Scheduling Problems

Traditionally, scheduling problems have been approached by translating the plant description into a MILP problem. This has been a popular approach because the user can manipulate and adjust the formulation as desired. Furthermore, there was no alternative since no higher-level software existed. Recently, Process Systems Enterprise Ltd., of Hammersmith Bridge, London, made available an integrated software package, gBSS. The benefits of this are that the user is not required to be familiar with the intricacies of MILP problems nor the mathematical formulation used by gBSS. However, the user is limited to the formulation used by gBSS and subsequently any limitations which exist in that formulation. In this section, the various software that is available for the solution of scheduling problems is evaluated.

2.5.0.1 Mathematical Modeling Software

AMPL and GAMS are mathematical modeling languages, which were developed to be able to solve large and complex mathematical programming models. They are designed such that the problem formulation is very similar to the mathematical representation, which simplifies the process of setting up and following of the programs. GAMS and AMPL have a similar construction. These languages act as an interface between the FORTRAN solvers they implement and the mathematical model. The syntax in these languages is such that the programs are easy to read and can be checked by people other than the modeller.

The modeling language GAMS (General Algebraic Modeling System) was developed at the World Bank to facilitate the solution of multi-sectorial economy wide models. AMPL (A Modeling Language for Mathematical Programming) was developed at AT&T Bell Laboratories for communication applications. Versions of both GAMS and AMPL can be obtained for mainframes, workstations and PC's. Both modeling languages can also be interfaced with a large number of solvers, such as CPLEX and MINOS, to solve linear, mixed-integer linear, nonlinear and mixed integer nonlinear programs.

GAMS is a high level language, which makes use of concise algebraic statements to describe mathematical models. A GAMS program consists of a series of statements, which declare the data, variables and the equations. These equations describe the relationships between the variables and data. GAMS allows for the compact representation of large and complex models, which make it easy both to understand and implement. Based on a particular type of problem (linear, non-linear, mixed integer *etc.*) GAMS calls on the corresponding adapted solver (such as MINOS for NLP) to solve the problem. AMPL has essentially the same features and capabilities as GAMS.

Programs written in GAMS and AMPL have a similar structure and a logical progression. The syntax in both languages is also fairly similar, although there are a number of subtle differences. GAMS provides a comprehensive output summary which makes it easy to detect errors. However, the displaying of results in GAMS is fairly rigid compared to AMPL, which is much more flexible in displaying the output that it produces. A major advantage that AMPL has over GAMS is that it has its own environment which allows post-solution data manipulation.

2.5.0.2 gBSS - An Integrated Scheduling Package

gBSS is an integrated software package developed by Process Systems Enterprise Ltd. This is commercial software which can be used for short-term scheduling, long-term campaign planning and the design of multipurpose batch/semi-continuous plants. Through the application of the STN representation, gBSS can handle many of the complexities that arise in practice. Some of these features are listed below.

- The user may specify maximum and minimum production requirements for individual products at different times over the planning horizon.
- Production recipes which involve the recycling, mixing and splitting of material between process steps may be implemented.
- A processing unit may be used as intermediate storage to simulate the NIS flowshop. All other flowshops can also be handled.

- Utilities such as steam and electricity and their availability may be specified and thus integrated into the problem formulation.

Equipment may be specified to perform a number of tasks over the scheduling horizon. Each equipment item is either characterized as an 'individual task' or a 'class of tasks'. For example an 'individual' task may be the second step in the manufacturing of product A, whereas a 'class of tasks' may be all tasks involving caustic reactions taking place at any stage of the product manufacture. In the allocation of equipment to tasks, gBSS ensures that only one task can be assigned to a suitable piece of equipment at any one time. The operation is non pre-emptive i.e. once a piece of equipment starts processing a task it cannot be interrupted. gBSS also ensures that at least one piece of equipment is assigned to each task so that the task is carried out. Batch units are characterized by their maximum capacities while semi-continuous and continuous plants are classified by their maximum processing rates.

The user is required to specify the following problem information in the form of 3 input files.

1. The process recipes for the products following the STN convention.
2. Resources available such as equipment and utilities.
3. The problem; short-term scheduling or campaign, the planning horizon, product delivery requirements and the particular solver to use (e.g. CPLEX, XPRESS *etc.*).

From the input files the package constructs a formal mathematical description from these problems. A procedure implemented by the package ensures that the solution obtained is optimal or that it is within specified margin of optimality. Solving the problem involves the determining of the order and timing of the operations for each item of equipment, the flow of the material through the plant and resource utilization. gBSS provides two output files; the statistics of the solution and graphical representation of the solution in the form of a Gantt chart.

2.6 Summary

Typically batch scheduling involves determining the optimal allocation of limited resources to carry out a set of tasks. The scheduling problem can be divided into two main categories based on the associated business conditions; namely short-term scheduling and campaign planning. Plants which are geared to satisfy individual customer demands and operate over time horizons of a couple of days to a week are classified as short-term scheduling problems.

Short-term planning involves the determination of a detailed schedule indicating the resource allocation and the sequence in which products are to be produced. Campaign planning on the other hand is more suited for plants operating under stable demand conditions producing a small subset of the potential products over long periods of time. Campaign planning involves the allocation of the available production time among various campaigns and the determination of the periodic structure of each campaign.

Three algorithms were presented for the short-term scheduling problem. The first algorithm makes use of recursive relations to establish a precedence order in the processing of products. However, a number of assumptions are made in the formulation, thus restricting its generality. The second and third formulations incorporated the STN representation into the problem formulation. The STN representation is quite flexible and results in the formulation of a general approach, allowing a better representation of practical situations. The second and third formulations differ in the method in which they handle the representation of time. The method presented by Shah *et al.* (1993a) is based on a discrete-time representation, while that presented by Ierapetritou and Floudas (1998a) implements a continuous-time representation.

The software that is commercially available to solve these types of problems can be categorized into two sections, general modeling languages (GAMS and AMPL) and scheduling-specific advanced languages (gBSS).

As pointed out in the section on on-line scheduling, there are a number of shortcomings to the use of off-line scheduling procedures to manage the operation of the

plant. The on-line scheduling problem basically deals with determining the deviations between the operation of the plant and the original predicted schedule and then modifying the schedule to reduce wait times of the products and the idle times of the machines.

Chapter 3

The Batch Plant

The focus of this chapter is on the scheduling of a batch plant. The plant is a scaled down version of the industrial problem, although it does include the characteristic features of the industrial plant. The first section provides a description of the layout and the operation of the scaled down plant. This is followed by the mathematical formulation that was implemented in order to determine the optimal schedule. A series of simulations conducted on a spreadsheet package are discussed next. The motivation for the spreadsheet simulations is twofold; firstly to show the non-triviality of the solution and also to highlight the major optimization variables. The final section details the optimal scheduling that was obtained via formulating the problem as a mixed-integer linear programming problem. A number of different scenarios are also included, which show the re-organization of the schedule in order to minimize the makespan, when faced with a change in the operating conditions. The associated computational aspects are also addressed in this section.

3.1 The Plant Description

Figure 3.1 is state-task network representation of the plant, where each task is represented by a rectangle and storage units are represented as circles. The STN framework has been slightly modified for clarification reasons. More specifically,

where a unit processes a number of tasks, the tasks are depicted as solid-lined rectangles and the units which process these tasks are represented as perforated-lined rectangles. The operation of the plant conforms to a multiproduct flowshop, where each product is processed through the same stages and in the same order. The only area where this is not the case is on unit 4 where the order in which tasks 4_1 and 4_2 take place is unknown.

The material available for processing is derived from four different sources, with each source having distinct characteristics. Depending on the current status of the plant, the amount of material in each source is normally sufficient to require the sub-division of the material into smaller lots. This is due to the fact that the maximum processing capacities of the units are constrained by an upper limit. The mass, number of lots each source is sub-divided into and the order of processing these lots are left as optimization variables.

The nomenclature used in the following description corresponds to Figure 3.1. Task 1 separates the material into 3 fractions, namely F_1, F_2 and F_3, which report to storage units $S2$, $S3$ and $S4$ respectively. The fractions into which the lot is sub-divided are dependent on the origin of the material, with each source having a unique composition. The material reporting to storage unit S2 is divided between units 2 and 3. Units 2 and 3 perform the same function, with unit 3 processing the material at a slightly slower rate. Hence the division of material between units 2 and 3 is left as a variable. The material from storage units S3 and S4 is processed sequentially on unit 4. The material from tasks 2, 3, 4_1 and 4_2 is then recombined in unit 5, which processes all the material from a particular lot simultaneously. Hence tasks 5_1, 5_2 and 5_3 must all happen concurrently for each lot. For simplicity where the task name corresponds to the unit name, such as task 2 which is processed on unit 2, these will be referred to interchangeably.

The operation of the plant is constrained by the fact that the inter-mixing of lots is forbidden, this is required for accurate accounting purposes after each stage. This constraint implies that if material is present in a storage unit, no further material may be added to that storage unit until the original material has been removed. A further complication is that material may not remain idle in a processing unit once

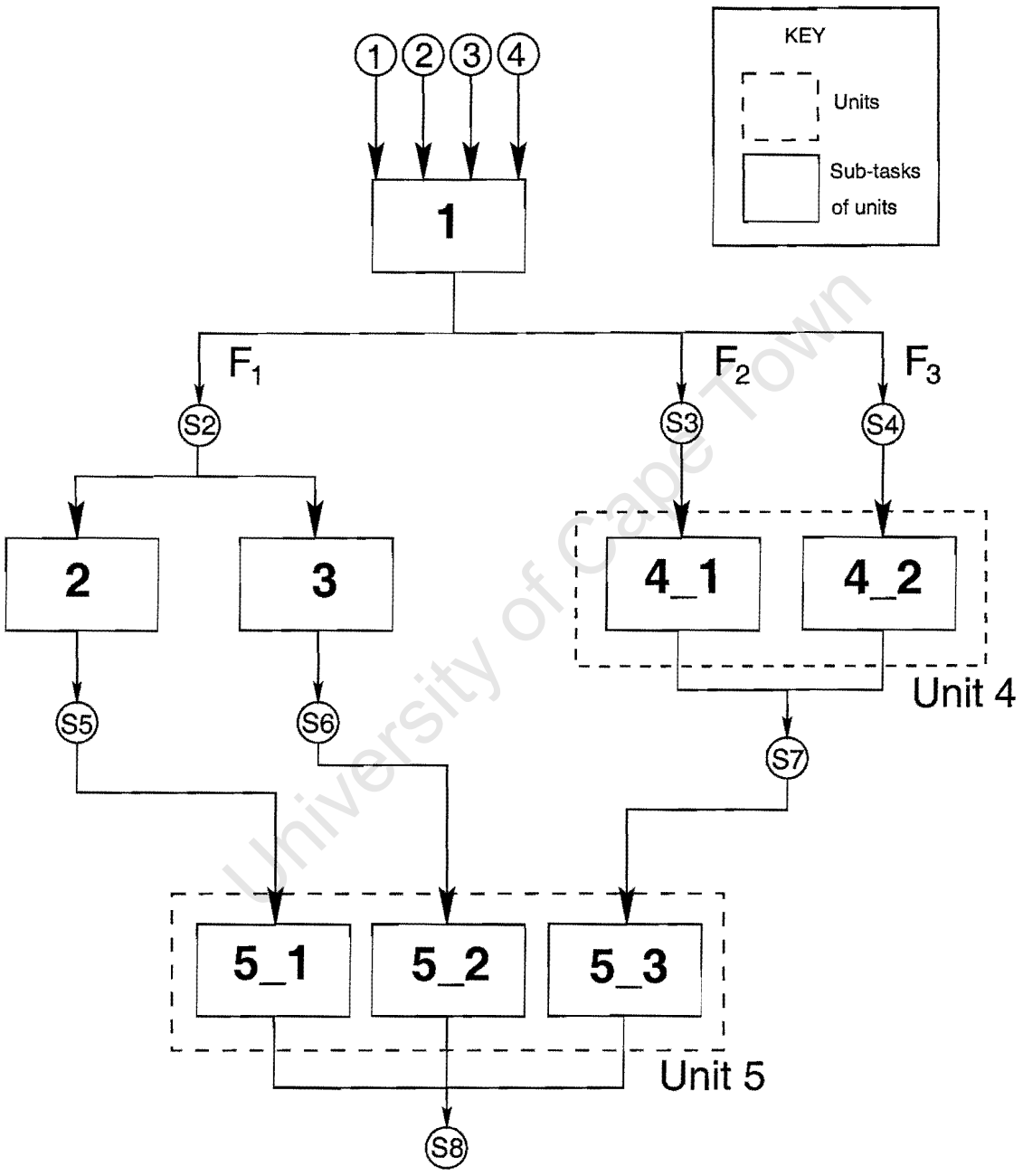


Figure 3.1: Modified STN diagram of the plant

processing is complete. Once a unit has completed processing a task, the products of that task are immediately moved to the downstream storage unit. Therefore task 1 of the subsequent lot cannot finish processing before tasks 2, 3, 4_1 and 4_2 of the present lot have been initiated. Furthermore tasks 2, 3, 4_1 and 4_2 cannot finish processing until the processing of tasks 5_1, 5_2 and 5_3 from the previous lot has begun processing. The result of these constraints is that the initiation of tasks may be delayed causing an overall increase in the makespan.

An interesting outcome of the no-mixing principal is that the order in which tasks 4_1 and 4_2 are scheduled has a definite impact on the makespan. Since there are no sequence dependent setup times associated with the processing of tasks 4_1 and 4_2, one would intuitively think the order in which the tasks 4_1 and 4_2 are processed is arbitrary. However, the combination of the following 3 factors makes the scheduling of tasks 4_1 and 4_2 nontrivial.

1. The processing times of tasks 4_1 and 4_2 are dependent on the mass of material being processed and are therefore a function of the particular lot being processed as well as the size of the lot.

2. Material from the upstream lot cannot finish processing on unit 1 until both tasks 4_1 and 4_2 have been initiated. In this case the task (either 4_1 or 4_2) with the smaller processing time being scheduled first would allow processing on unit 1 to start sooner, since the second task on unit 4 would also be able to start sooner.

3. Neither task 4_1 nor 4_2 can finish processing until the downstream lot has started processing on unit 5. If there is a bottleneck at unit 5, it would be preferable to process the task with the longer processing time first, thereby delaying the finish of the first task.

From points 2 and 3 above, it is clear that there may be cases where the scheduling of tasks on unit 4 may be beneficial for the upstream lot at the expense of the present lot or vice versa.

The object of this work is to determine the optimal schedule in which all material is processed in the minimum time, subject to the constraints described above. The form of the objective function is as follows.

$$Z = \text{Makespan} = T^f_{i'j'kn} \tag{3.1}$$

where $T^f_{i'j'kn}$ is the completion time of the last task i' on the last unit j' in the time horizon.

Tables 3.1, 3.2, 3.3 and 3.4 detail the operating conditions of the plant. More specifically, Table 3.1 provides the initial mass of material in each source that is required to undergo processing. Table 3.2 details the processing times associated with each task. The dead time refers to the portion of the processing time which is independent of the amount of material processed, this takes into account setup and transfer times for each task. The processing time of each task (other than those conducted on unit 5) is also dependent on the amount of material that is processed in that unit. The composition of the material from each source is presented in Table 3.3, where F_1, F_2 and F_3 are the fractions into which the material is separated and correspond to Figure 3.1. Table 3.4 details the capacity of the plant units.

Table 3.1: Initial mass of material in each source

Source Origin	**Total Mass** $[kg]$
1	65
2	91
3	45
4	73

Table 3.2: Processing times of tasks

Task	**Dead Time** $[min]$	**Processing Rate** $\left[\frac{min}{kg}\right]$
1	20	3.20
2	10	18.0
3	10	16.0
4_1	15	8.00
4_2	10	10.00
5_1/2/3	170	-

Table 3.3: Fractional composition of source material

	Source 1	**Source 2**	**Source 3**	**Source 4**
$\mathbf{F}_1$	0.2	0.8	0.6	0.1
$\mathbf{F}_2$	0.3	0.1	0.3	0.6
$\mathbf{F}_3$	0.5	0.1	0.1	0.3

Table 3.4: Unit capacities

Unit	**Minimum** $[kg]$	**Maximum** $[kg]$
1	10.0	50.0
2	1.0	40.0
3	1.0	40.0
4	1.0	40.0
5	10.0	50.0

3.2 Formulation

The formulation of the problem into a mathematical model was based on the continuous-time formulation proposed by Ierapetritou and Floudas (1998a), which was presented in Section 2.3.3 of the literature review. To avoid repetition, only the aspects that differ between the implemented formulation and that presented in the literature review are addressed in this section. The decision for selecting this formulation was made based on the following factors:

- The continuous-time formulation is preferred over the discretized-time formulations for two reasons. Firstly, the time horizon is a continuum and it is not necessary to approximate the processing times of tasks by using discrete intervals. Most importantly, the formulation presented by Ierapetritou and Floudas (1998a) can be easily modified to encompass the fact that processing times are dependent on the amount of material being processed, for the case where the amount of material in each lot is also a variable.
- It has been shown (Ierapetritou and Floudas, 1998a,b, 1999) that for the selection of problems available in the literature that this formulation, when compared to other continuous-time formulations, is superior with respect to both the generality of the formulation and its computational efficiency.

3.2.1 General Alterations

An extra index, k, has been added to all variables except the storage units, where k refers to the source of the material. For example the variables wv_{in} and T^s_{ijn} become wv_{ikn} and T^s_{ijkn} respectively. This change was implemented to aid the differentiation of sources in the formulation. Each source has distinct properties and therefore lots can easily be associated with the sources from which they originate. Alternatively, all lots regardless of their origin could have been given a distinct variable name. However, since the number of lots into which each source will be sub-divided is not known *a priori* it is is best to use the method of distinguishing material by its origin and leave the number of lots as a variable.

The objective function that was used in all cases is the makespan, which is minimized. Obviously, the smallest makespan is obtained if no tasks take place, for this reason it is necessary to add hard constraints to the formulation which ensure that all the material is processed. This is achieved through the implementation of equation (3.2).

$$\sum_{s \in S_d} \sum_{n \in N} d_{skn} \geq ST^{IN}_{k,n_0} \qquad \forall\, k \in K \qquad (3.2)$$

where:

ST^{IN}_{k,n_0} is the initial amount of material available in each source, k.

d_{skn} is the amount of material originating from source k that is removed from the system at event point n.

S_d are the states from which material is removed.

3.2.2 Lot-size dependent Processing Times

Processing times of all tasks except those carried out on unit 5 (see Figure 3.1) are dependent on the mass of material processed in that task. Since the mass of material in each task is a variable, the processing times cannot be determined *a priori*. Kondili *et al.* (1993) proposed the method of using a piecewise function approximation for batch-size dependent processing times, which was detailed in Chapter 2, Section 2.3.2.1. However, the addition of this new set of binary variables in the piecewise approximation results in an overall increase in the computational complexity of the problem. As mentioned previously, the addition of each extra binary variable can as much as double the size of the required search tree in the branch and bound algorithm. Another drawback of using a piecewise approximation is that the level of approximation in the problem is further increased. In some instances this may deviate from the real situation to a point where a optimal solution may in fact be sub-optimal for the actual situation.

To determine the duration of tasks, Ierapetritou and Floudas (1998a) proposed a constraint (see equation (2.22) on page 27) which uses the size of the batch to determine the degree of variability in the processing time of that task. A slight

modification to the application of this constraint allows the duration of tasks to be determined by dividing the amount of material in that task by the processing rate. Hence the task duration is not required to be known *a priori*. Equation (3.3) shows the form of the constraint. In this application the variable λ_{ij} refers to the processing rate, whereas Ierapetritou and Floudas (1998a) use the term β_{ij} (variable processing time constant) in its place.

$$T^f_{ijkn} = T^s_{ijkn} + \alpha_{ij} wv_{ikn} + \lambda_{ij} B_{ijkn} \qquad \forall\, i \in I,\ j \in J_i,\ k \in K,\ n \in N \tag{3.3}$$

where:

T^s_{ijkn}, T^f_{ijkn} are the start and finish times of tasks, respectively.

α_{ij} is the dead processing time and is associated with the task binary variable wv_{ikn}.

λ_{ij} is the inverse of the processing rate $\left[\frac{time}{mass}\right]$ and is associated with the task-mass variable B_{ijkn}.

The handling of batch-size processing times in the continuous-time formulation is more accurate and does not increase the number of integer variables in the formulation; it does however have a number of computational drawbacks.

With the objective being to minimize the makespan, the relaxed solution tends to guide the integer variables away from integrality in order minimize the processing times of tasks. The result is that many tasks are partially allocated to each unit in the relaxed solution. When integrality is imposed not all products can be produced in the same amounts as determined in the relaxed solution since units can only be allocated to the processing of a single task at a time. The consequence is that integral solutions are only found very deep down in the tree-search after the investigation of many nodes. As discussed in the literature review, if an integer solution is found at a node, further branching on that node will not produce a better solution. It follows then that the sooner the integral nodes are located in the tree search the fewer the number of nodes requiring investigation, since less branching has taken place. Locating integral nodes deep in the search tree impacts the required computational effort in two ways. Firstly, the time it takes to locate an integer solution is increased due to the deep search required. Secondly, the deeper

the search progresses into the tree the more branching occurs and hence the more nodes that need to be investigated.

Furthermore, the inclusion of the batch-size variable in the computation of the processing time of tasks, tends to increase the complexity of the computations at each relaxed sub-problem. It is suspected that this is due to the fact that the processing time and batch size are inter-dependent, thus complicating the algorithmic search. The overall result is that solution cost per LP is increased. In addition to this, using the batch-size as a variable in the computation of the processing time increases the integrality gap by loosening the LP relaxation of the MILP, since there is more flexibility in the system.

A comparison of the piecewise approximation method versus the use of the batch size as a continuous variable was carried out. This comparison was by no means exhaustive and its sole purpose was to determine if there was any advantage in discretizing the batch size as opposed to keeping it as a continuous variable. The outcome of the study was that the continuous method required far less computational effort in the particular examples that were investigated. However, it should be noted that this could have been due to the number of discrete values used in the approximation of the batch size and the particular problem under investigation. Furthermore, these comparisons were both carried out using the continuous-time horizon formulation, thus no comparison was made between the piecewise approximation implemented with a discrete-time formulation.

3.2.3 Preventing the Inter-mixing of Lots

A particular characteristic of the plant is that the inter-mixing of lots is forbidden. Once a lot begins processing in unit 1 it may not combine with any other lot, including lots derived from the same source. This provides a further complexity in formulating this problem into a mathematical model. As discussed in the literature review, Section 2.3.2.1, Kondili *et al.* (1993) proposed a method to prevent the intermixing of material in a multiproduct plant, where different tasks output to the same storage units. They added a storage task which removes material from

the storage unit and then produces the same amount during the next interval. By using integer variables it is possible to constrain the use of the storage tasks and hence the storage units themselves, to the products of a single task at any one time. In the particular problem being investigated the number of storage tasks required would be equal to the number of units and hence the number of integer variables added to the problem would be significant. An alternative method of preventing the intermixing of lots is therefore proposed.

In order to prevent material from different lots from being combined in a storage unit the constraint given in equation (3.4) is added.

$$wv_{ikn} \leq 1 - \frac{ST_{s_{i+1},n}}{ST^{max}_{s_{i+1}}} \qquad \forall\, i \in I_s,\, k \in K,\, n \in N,\, s \in S \tag{3.4}$$

where:

$ST_{s_{i+1},n}$ is the mass of material in storage unit s which is directly downstream from the unit in which task i occurs.

$ST^{max}_{s_{i+1}}$ is a parameter representing the maximum capacity of storage unit s, directly downstream from the unit in which task i occurs.

I_s is defined as the set of tasks which can produce state s.

This constraint is particular to the formulation proposed by Ierapetritou and Floudas (1998a). In this formulation, material is produced by a task at event point n and is only added to the storage unit at the next event point. Equation (3.4) simply states that if a task takes place at an event point then at that event point the downstream storage unit must be empty (*i.e.* $ST_{s_{i+1},n} = 0$) else the task cannot be carried out. According to equation (2.20) (on page 26), the mass of material undergoing task i at event point n will only be added to the storage unit event point $n+1$. At event point $n+1$ the binary variable wv_{ikn} is once again be equal to zero. Therefore the left-hand side of equation (3.4) is equal to zero and the amount of material from that task is constrained by the maximum capacity of the storage unit ($ST^{max}_{s_{i+1}}$).

To ensure this always holds (*i.e.* no portion of the material bypasses the storage unit) and that no material is left in the storage unit when this material is transferred

to the downstream unit (an industrial operational requirement) equations (3.5) and (3.6) are incorporated in the formulation.

$$wv_{ikn} \leq \sum_{i' \in I_{Di}} wv_{i',k,n+1} \qquad \forall\, i \in I,\ k \in K,\ n \in N \tag{3.5}$$

where I_{Di} is the set of tasks suitable for downstream processing of the products of task i; i' is a task in the set I_{Di}.

Equation (3.5) ensures that if a task takes place at event point n, then at least one of the tasks in the set I_{Di} will process the products of task i at the next event point. The material balance and allocation constraints determine whether the product material of task i is split between a number of tasks in the set I_{Di} or processed as a single task.

Equation (3.6) ensures that the amount of material processed in the downstream task at event point $n+1$ cannot be less than the portion of material, $\rho^p_{si} B_{ijkn}$, which task i produced into state s at event point n.

$$\rho^p_{si} B_{ijkn} \leq B_{i',j',k,n+1} + V^{max}_{i'j'}(1 - wv_{i',k,n+1})$$
$$\forall\, s \in S,\ i \in I,\ i' \in I_{Di},\ j \in J_i,\ j' \in J_{Di},\ k \in K,\ n \in N \tag{3.6}$$

where:

ρ^p_{si} is the proportion of material from task i that reports to task $i+1$ via state s and is consistent with equation (2.20) (on page 26).

J_{Di} is the set of units, downstream from unit j, upon which the set of tasks I_{Di} are processed.

$V^{max}_{i'j'}$ is the maximum capacity of task i' when processed on unit j'.

The term, $V^{max}_{i'j'}(1 - wv_{i',k,n+1})$ is included in equation (3.6) since not all tasks in the set I_{Di} need occur in the processing sequence. This term ensures that the constraint is trivially satisfied if task i' does not take place at event point $n+1$.

Equations (3.5) and (3.6) both provide computational improvements to the formulation since the values of both the allocation and batch-size variables are constrained to the values of the previous task.

For units which operate in parallel and where the division of material between these units is left as a variable, equation (3.6) is adapted to the form of equation (3.7). However, this is only applicable to instances where it is known that the set of parallel tasks *will* occur at the next event point, since the term $V^{max}_{i'j'}(1 - wv_{i',k,n+1})$ cannot be included due to the summation.

$$\rho^{p}_{si} B_{ijkn} \leq \sum_{i' \in I_{Pi}} (B_{i',j',k,n+1}) \qquad \forall\, s \in S,\ i \in I,\ j \in J_i,\ j' \in J_{Pi},\ k \in K,\ n \in N \tag{3.7}$$

where:

I_{Pi} is introduced to define the set of parallel tasks which are suitable for processing the products of task i.

J_{Pi} is the set of units on which the set of tasks I_{Pi} are processed.

3.2.4 Recombination of Material

Another characteristic of the plant is that all the material from a particular lot must be recombined and processed simultaneously in unit 5. Therefore tasks 5_1, 5_2 and 5_3 must all happen simultaneously and hence, have the same start and finish times. For convenience, these tasks have been included as three distinct tasks in the formulation, however, in reality they form a single task. To accomplish these operational policies the following constraints are included in the formulation.

$$\sum_{i \in I^{c}_{j}} \sum_{k \in K} wv_{ikn} = \pi \times yv_{jn} \qquad \forall\, j \in J,\ n \in N \tag{3.8}$$

where:

I^{c}_{j} is the subset of tasks that must occur concurrently on unit j.

π is the number of tasks in the set I^{c}_{j}.

Note that the binary variable yv_{jn} has no index, k, relating it to the source of the material, thus ensuring only a single task occurs at each event point. Equation (3.8) ensures the consistency of the mass balance in the formulation. However, further constraints are necessary to ensure that the timing of the tasks is correct, *viz.* equations (3.9) and (3.10).

$$T^s_{ijkn} = T^s_{i'jkn} \qquad \forall\, i \in I^c_j,\ i \neq i',\ j \in J_i,\ k \in K,\ n \in N \tag{3.9}$$

$$T^f_{ijkn} = T^f_{i'jkn} \qquad \forall\, i \in I^c_j,\ i \neq i',\ j \in J_i,\ k \in K,\ n \in N \tag{3.10}$$

These constraints are in addition to the other timing constraints presented in the formulation and ensure that these tasks all have the same start and finish times, thus ensuring that the recombination task will not begin until all the sub-tasks are ready for processing. It should be noted that these constraints are only implemented for the set of tasks which occur concurrently on unit 5.

3.2.5 Computational Issues

3.2.5.1 Improvements

Margin of Optimality A large proportion of the computational effort required in finding the solution using the branch and bound method is often due to the verification of an optimum solution as opposed to finding it. The computational performance can thus be greatly enhanced by specifying that the final solution is only required to be optimal to within a small integrality gap, of say 5%. In this work it was often the case that the solutions terminate to within a smaller integrality gap than specified, as the final iteration takes the solution past the desired level of accuracy. In most cases the last integer solution terminated with an integrality gap of 2-4%. However, it was found that ensuring the solution was the best possible (*i.e.* no integrality gap at termination) sometimes took 3 times longer than the rest of the solution process. In all the test work it was found that there was little to no improvement in the solution when forcing the problem solve to within a an integrality gap of zero as opposed allowing a small margin.

Removal of Superfluous Binary Variables In cases where at most a single task may occur on a unit at an event point (*i.e.* tasks do not occur concurrently on the same unit), the inclusion of the binary variables, yv_{jn}, which are used to assign the utilization of units (see equation (2.17) on page 25) can be substituted into equations (2.23), (2.26) and (2.27) and thus eliminated. The allocation constraints (equation (2.17)) can thus be altered, *viz* equation (3.11). This alteration is permitted by the definition of yv_{jn} and wv_{ikn} as binary variables, since they may only take on a value of either 0 or 1. Furthermore, it is not necessary to explicitly control unit allocations because the summation of the wv_{ikn} variables is carried out over the subset of tasks (I_j) which can be processed on unit j.

In this case the set of constraints can be modified to the following.

$$\sum_{i \in I_j} \sum_{k \in K} wv_{ikn} \leq 1 \qquad \forall\, j \in J,\ n \in N \qquad (3.11)$$

Thus a number of integer variables are removed from the problem. In the case where multiple task must occur simultaneously, such as on unit 5, it is necessary to implement the allocation constraints in the form of equation (3.8) (as described in Section 3.2.4 above), where either none of the tasks or all 3 tasks must occur at an event point.

Assigning Tasks to Event Points Due to the multiproduct nature of the plant it is possible to add a set of constraints that improve the computational efficiency of the formulation. Since the sequence of tasks follows a regular pattern it is possible to assign a set of tasks to an event point (*viz.* equation (3.12)). This in effect ensures that a unit processes one of the subset of tasks that may be processed on that unit, at a particular event point.

$$\sum_{i \in I_j} \sum_{k \in K} wv_{ikn} = 1 \qquad \forall\, j \in J,\ n \in n_l, \qquad (3.12)$$

where:

n_l is set of event points at which task i is active.

Equation (3.12) simply states that one of the tasks suitable for processing on that unit must occur at the designated event point. For the problem described earlier in the chapter, a product requires 4 unique event points to complete processing through the plant. Task 1 will take place at the first event point, tasks 2, 3 and one of the tasks on unit 4 will take place at the second event point, the other task on unit 4 is carried out at the next event point and the last task will take place at the fourth event point. Note that tasks 2, 3 and one of tasks 4_1 and 4_2 may all take place at the same event point, since they occur on different units. This is shown in Figure 3.2, where the markers represent the processing of task i at event point n (*i.e.* $wv_{in} = 1$).

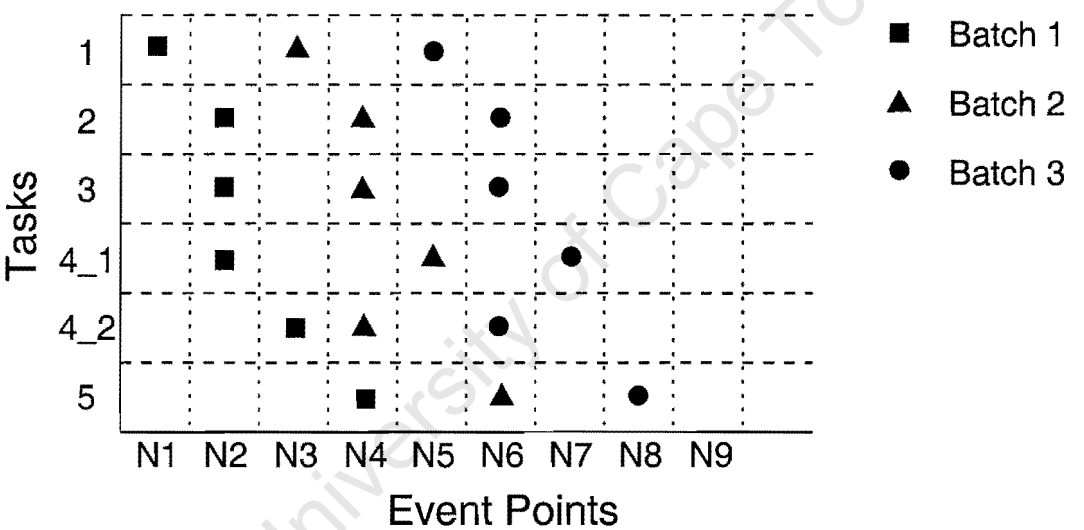


Figure 3.2: Example of event point assignment

Due to the fact that event points are independent of time, it is only necessary to ensure that at most one task is carried out per unit at each event point and that sequential tasks occur on sequential event points. In so doing resource-task allocation conflicts are prevented by the timing constraints in the formulation.

This point is illustrated in Figure 3.2, where task 5 from the first batch and task 4_2 from the subsequent batch are both processed at event point N_4. The fact that these tasks occur at the same event point does not cause any conflicts, since the initiation of task 5 is only dependent on the finish times of the tasks 2, 3, 4_1, 4_2

and 5 at previous event points. Thus the flexibility of the problem is not changed by allowing the initial tasks in the processing sequence of a batch to be active at the same event point as the final tasks of the previous batch as long as these tasks do not require the use of the same unit. The same end result would be achieved by dedicating a set of event points to processing of each batch. However, the benefit of allowing multiple non-conflicting tasks to occur at the same event point is that the total number of event points required in the formulation is reduced and hence the number of binary variables in the formulation is also reduced.

Due to the multiproduct operation of the plant, the allocation of tasks to units happens in a repetitive pattern (except the tasks on unit 4). Thus unit 1 will process a task at the first event point and then again at every second event point after that in order to ensure that the downstream tasks occur at the next event point (*viz.* equation (3.13)).

$$\sum_{k \in K} wv_{1,k,n_l} = 1 \qquad for\ n_l = n_1 + 2n \qquad (3.13)$$

The division of material into lots is a variable and thus so is the total number of lots required to process the available material. Subsequently, the number of lots that will be required to process all material, as determined by optimal scheduling, is not known *a priori* . Thus it is important when assigning sets of tasks to event points that these assignments do not require the processing of more than the minimum number of lots than is necessary to process all available material. This necessitates the determination of the minimum number of lots that may be processed for a particular set of input conditions. The minimum number of lots in each source is determined by dividing the total amount of material in each source by the respective maximum lot size of that source. All units have different capacity limitations and the division of material between tasks is dependent on the origin of the material, therefore it is necessary to determine the maximum lot size for each source.

Minimize the Sum of the Binary Variables Ierapetritou and Floudas (1998b) suggest the inclusion a penalty term in the objective function, which is comprised of the summation of the task binary variables; $\sum_{i,k,n} (P \times wv_{ikn})$, where P is a penalty parameter. This term penalizes the utilization of units and hence improves the computational performance of the solution, without influencing the schedule production in the case where products demands are represented as hard constraints (Ierapetritou and Floudas, 1998b).

3.2.5.2 Factors Complicating the Solution Procedure

There are a number of factors associated with this type of problem which degrade the solution procedure. Some of these factors are typical of most scheduling problems while others are unique to the plant being studied. The first aspect is that the mathematical formulation of scheduling problems results in a large number of binary variables, creating a potentially large branch and bound tree-search. However, Shah *et al.* (1993b) show that the number of binary variables in the formulation is not necessarily the best guide to determining the effort required in obtaining a solution.

Another useful indicator in determining the computational effort required is the integrality gap (which is the difference between the integer solution and the relaxed LP). In general, the greater the integrality gap the larger the search time (Yee and Shah, 1998). This is an intuitive result since the larger the difference, the greater the likelihood of more solutions existing between the final integer and initial relaxed solutions and hence the more nodes requiring investigation. A large integrality gap is found in problems which require frequent re-use of units and where more that one item of equipment is available for the same task. Furthermore, the use of a continuous-time horizon is reported to exhibit larger integrality gaps compared to those formulations which make use of a discretized-time horizon (Shah *et al.*, 1993b; Schilling and Pantelides, 1996; Yee and Shah, 1998).

As discussed in Section 3.2.2, the inclusion of units whose processing times are dependent on the size of the lots it processes complicates the solution procedure. The

plant being studied includes parallel units (namely units 2 and 3) and can therefore be categorized as having multiple units available for the same task. The inclusion of tasks which are processed in series (tasks 4_1 and 4_2) also tends to increase the integrality gap, because the relaxed solution partially assigns these tasks so that they occur simultaneously. Another aspect of the problem under investigation is that the optimal solutions are not unique. This can result in the solution process continuing long after the best solution has been found, as an exhaustive tree-search is carried out in an attempt to guarantee that the integer solution is indeed optimal.

3.2.5.3 Brief Description of Solver Options

As with most commercial optimization software, CPLEX allows the user to change the default solver specifications, such as the algorithm selection, scaling and so forth, thus allowing the user to exploit certain characteristics of the problem under investigation. Due to the large number of possible options that can be implemented, only the options that were explored and appeared to affect the computational speed of solution will be discussed here. The manner in which these options influence the computational efficiency depends on the problem itself and the combination of these options. Therefore the following discussion is of a qualitative nature. The following commands are GAMS commands that override the default CPLEX options. These commands can be directly implemented with CPLEX although the syntax differs. These options are valid for CPLEX version 6.6 and GAMS version 2.50E.

scaind This setting controls the problem matrix scaling. The options available are standard scaling, modified scaling or no scaling. For problems which are poorly conditioned it is recommended that the modified scaling option is used. This option provides more aggressive scaling which can be beneficial on some problems. It was found that this was the case in many of the scenarios that were carried out.

subalg Subalg is used to select the algorithm used to solve linear sub-problems at each node. CPLEX offers the primal simplex, the dual simplex, and the barrier method as well as simplex-barrier combinations. The

default option is the dual simplex method which is recommended for the majority of problems. The use of the barrier method is recommended in the case of large (over a thousand rows or columns), sparse (relatively few non-zeros) problems. All the problems investigated in this work were large, with over 6000 rows and over 1500 columns. The number of non-zeros in this matrix was in the order of 2-5% of the total number of elements, depending on the particular example. Consequently the problem matrix is extremely sparse. Contrary to these recommendations it was found that the dual simplex algorithm outperformed all other methods.

dpriind This setting is used to manipulate the pricing strategy for the dual simplex algorithm. The available options are either standard pricing or 3 variations of steepest-edge pricing. In cases where the subproblems at each node are taking many iterations to solve, steepest-edge pricing is preferred over standard pricing, although standard pricing is reported to be more stable. In the scenarios that were tested it was found that the steepest-edge pricing was superior. However, one drawback of the steepest-edge pricing option is that is seems to be extremely heavy on resources and therefore its use is limited to problems which solve before a large search-tree is generated. For example, in some of the test work it was found that the node-file size grew to over 1 Gb after the investigation of more than 5000 nodes. This requires CPLEX to write the node-file to a swap-file on disk which retards the computational efficiency. Using standard pricing on the same problems, the convergence to the solution required the investigation of more nodes in total; however the node-file size generally remained below 100 Mb, which was small enough to reside in the physical memory.

varsel Varsel allows the user to set the rules used in the selection of the variable for branching at each node. It is possible to select the branching variable based on minimum infeasibility, maximum infeasibility, pseudo costs or to specify strong branching. The default is to allow CPLEX to select the branching variable automatically. However, in order to

improve performance it is suggested that branching based on pseudo costs be used. Pseudo costs are a measure of the expected objective function change, for fixing the candidate variables to 0 and 1. In cases where obtaining an integral solution takes a considerable time or the first solution is far from the best bound value, using strong branching is advised. Strong branching requires more computation for each branch but in many cases obtains very good solutions in fewer nodes and less total time. However, it was found that in some cases the total solution time was smallest when the branch-variable selection was based on pseudo costs.

nodesel This option is used to select the next node to process when backtracking up a branch. The available options are to choose the most recently created node, a best-bound search or a best-estimate search. The best bound search selects the node with the best objective function based on the relaxed LPs. The best-estimate search selects the node with the best estimate of the integer objective value that would be obtained once all infeasibilities have been removed. Through trial and error it was determined that the best-estimate search provided the best results, obtaining the final solution in the shortest time.

brdir Brdir can be used to decide which branch direction should be taken first by the selected variable. Branching can either be determined automatically, *up* or *down*. Branching *up* sets the selected branching variable to be one, while branching *down* sets the branching variable to zero. A particular aspect which may be taken advantage of in scheduling problems is that the sum of binary variables must be less than or equal to one. Therefore setting the branch-first parameter to *up* sets the selected branching variable to one, which then forces the rest of the variables in the summation constraint to zero. This eliminates all infeasibilities in the constraint, whereas branching *down* only causes one infeasibility to be eliminated. Therefore the branching direction was set *up* for all problems.

3.3 Scenarios

A preliminary study on the flexibility present in the system was carried out using a spreadsheet package. The motivation behind this study was to ensure that the scheduling of material through the plant was non-trivial. Upon completion of the preliminary study the problem was formulated into a mathematical description using an MILP formulation. The formulation is based on the continuous-time formulation proposed by Ierapetritou and Floudas (1998a). The initial condition of the plant for all scenarios is presented in Section 3.1. In scenarios where the conditions differ to those stated in Section 3.1, the difference is noted in the presentation of that scenario.

3.3.1 Preliminary Study - Determination of Flexibility

To determine the scope for optimization, a number of simulations were conducted using a spreadsheet package. The timing of the tasks was determined simply by using logic-based formulae, to ensure that there was no inter-mixing of lots and that all other operational constraints were obeyed. Units 2 and 3 perform the same task although at slightly different rates. The division of material between the parallel units was thus determined using a linear solver to ensure that the processing times of tasks 2 and 3 were the same. Splitting the material in this manner, minimizes the combined effect of the bottleneck at these units.

The major optimization variables are the relative sizes into which sources are subdivided and the order in which the lots are processed. Another area of flexibility is the sequence in which tasks 4_1 and 4_2 are processed on unit 4. Therefore 4 different scenarios are presented here; the base case, changing the order of processing, changing the sub-division of material into lots and the order of processing tasks 4_1 and 4_2.

3.3.1.1 Scenario 1.1: Base Case

As a base case it was decided to divide the mass of material in each source equally between the number of lots in that source. The maximum capacity of the units restrict the size of the lots and some sources required multiple lots to process all material. It was decided to use the minimum number of lots for all simulation scenarios, an issue which is addressed in the following section. It is expected that this is what the optimal solution will specify due to the fact that there are dead times associated with all tasks and dead times tend to penalize the use of extra lots. Table 3.5 details the conditions (order of processing and lot size) used in the base case scenario. The lots are referred to by the source from which they originate. Sources 1, 2 and 4 are divided into 2 lots each and source 3 is processed as a single lot, making up a total of 7 lots.

Table 3.5: Lot division and order of processing for the base case

Order of Processing	1	2	3	4	5	6	7	Total
Source Origin	1	1	2	2	3	4	4	
Mass of Lot $[kg]$	32.5	32.5	45.5	45.5	45.0	36.5	36.5	274.0

The makespan for the base case scenario is 1964 minutes.

Gantt charts are the most common form of displaying the results of scheduling problems. For the reader who is unfamiliar with such charts an explanation is given. The tasks are listed on the vertical axis, starting with task 1 at the top and progressing to task 5 at the bottom, which corresponds the order in which the tasks are carried out. In all the Gantt charts presented, tasks 5_1, 5_2 and 5_3 will be represented as a single task, namely task 5. Since these tasks must always occur simultaneously, presenting them separately would be unnecessarily repetitive. The horizontal axis is the time axis, which has units of minutes. The horizontal bars represent the duration of the tasks.

Each lot/batch has a distinct shading to differentiate it from other tasks and allows the easy tracking of the batch's progress through the plant. To allow easier distinction of the material from different sources, lots from the same source have been

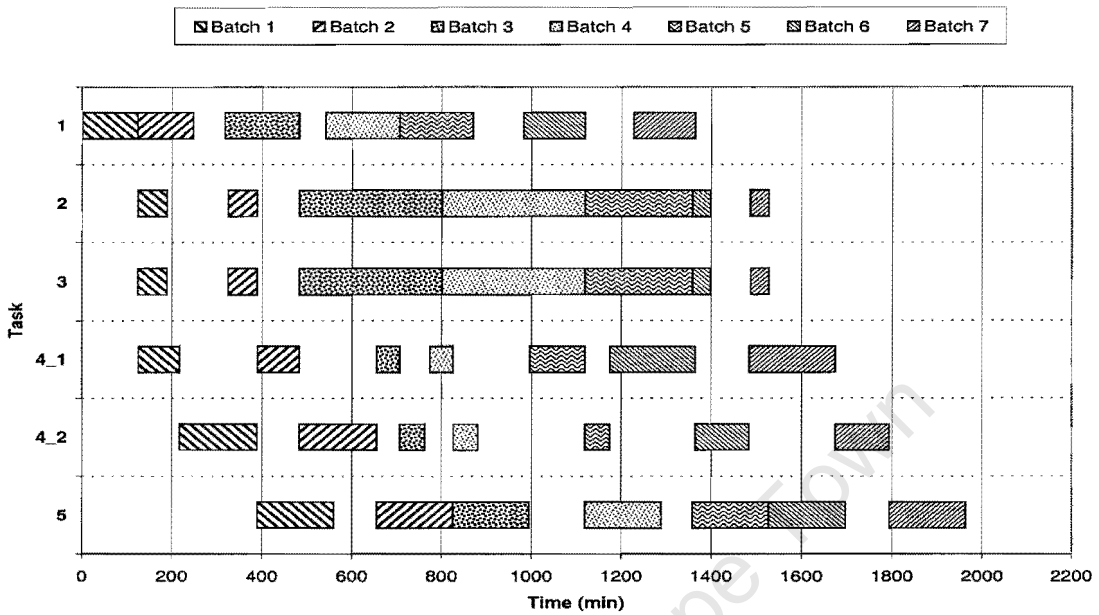
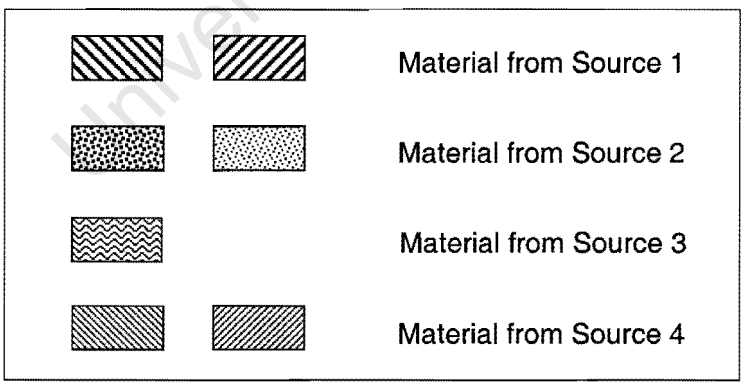


Figure 3.3: Gantt Chart - Base case scenario

shaded in a similar pattern, as shown in the legend below. This legend is consistent throughout this thesis.



The reader's attention is drawn the following features of the plant which are highlighted in Figure 3.3.

- A Task may not begin processing until its precursory task has completed processing. Therefore tasks 2, 3, 4_1 and 4_2 may only begin once task 1 has

finished. Similarly the set of tasks on unit 5 must wait for the upstream tasks; 2, 3, 4_1 and 4_2, to finish processing. This restriction is fairly obvious since the products of a particular task will not be ready for the downstream task until processing has been completed.

- A task must wait for the unit to be free before it can begin processing on that unit. Once again this restriction is intuitive and evidence can be seen of this in Figure 3.3 where the processing of tasks 2 and 3 of the fourth batch is delayed by the same tasks from the third batch.

- The downstream storage unit must be free before a task finishes processing. This restriction is borne out of the fact that no inter-mixing of batches is allowed. In Figure 3.3 the initiation of task 1 for batches 3, 4, 6 and 7 is delayed to ensure that these tasks do not finish processing until tasks 2, 3, 4_1 and 4_2 of the previous batch have all begun processing. This feature is also prevalent further downstream in the plant, where the initiation of task 4_1 of the fourth batch is delayed so that it does not finish until task 5 from the previous batch has begun processing.

3.3.1.2 Scenario 1.2: Change in the Sequence

The aim of this scenario is to show that the order in which lots are processed influences the makespan. All conditions are the same as those used in the base case scenario, except the order in which the lots are processed. The order of processing is reported in Table 3.6.

Table 3.6: Lot division and order of processing for Scenario 1.2

Order of Processing	1	2	3	4	5	6	7	Total
Source Origin	1	2	3	4	1	2	4	
Mass of Lot $[kg]$	32.5	45.5	45.0	36.5	32.5	45.5	36.5	274.0

Figure 3.4 presents the results of this scenario in the form of a Gantt chart. As can be noted in the comparison of Scenario 1.1 and this scenario (Figures 3.3 and 3.4, respectively), the alternate sequencing of lots from different sources in this scenario

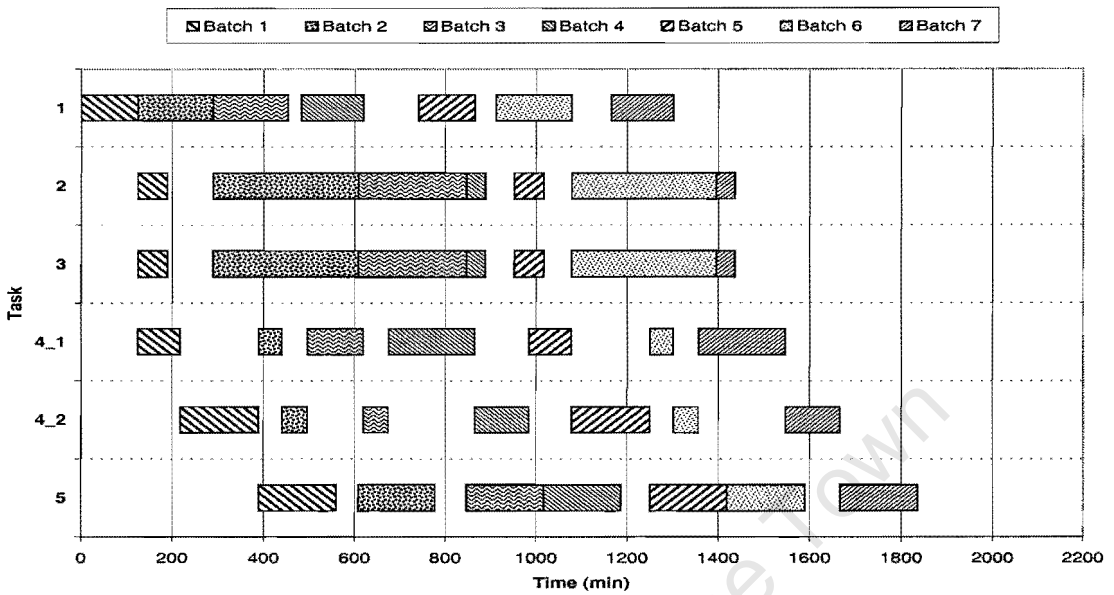


Figure 3.4: Gantt Chart - Change in the order of processing lots

yields a superior schedule. In the base case scenario, the initial bottleneck on unit 4 is followed by bottlenecking in units 2 and 3. While in this scenario, the extent of initial bottlenecking on unit 4 is reduced and the bottlenecking at units 2 and 3 is also alleviated somewhat. The overall result is a reduction in the makespan, which is 1836 minutes in this scenario compared to the makespan of 1964 minutes in the base case scenario. This shows an improvement of 2 hours over a 32 hour time period which is approximately a 6% reduction. From this scenario, it can be concluded that the order in which the lots are sequenced has a significant impact on the makespan.

3.3.1.3 Scenario 1.3: Change in Sub-division of Lots

This scenario aims to show that the relative division of material into sub-lots affects the makespan. The order of processing the batches though the plant is the same as that used for the base case. The size of the individual lots for each source have been altered, although the total mass of material in each source is consistent with that of the base case scenario. The method used to divide the sources into lots was

to assign the largest possible mass of material, which is restricted by the maximum operational capacity of the units, to one lot and put the remaining material into the other lot. The individual lot sizes and their respective order of processing are reported in Table 3.7.

Table 3.7: Lot division and order of processing for Scenario 1.3

Order of Processing	1	2	3	4	5	6	7	Total
Source Origin	1	1	2	2	3	4	4	
Mass of Lot $[kg]$	15.0	50.0	41.0	50.0	45.0	23.0	50.0	274.0

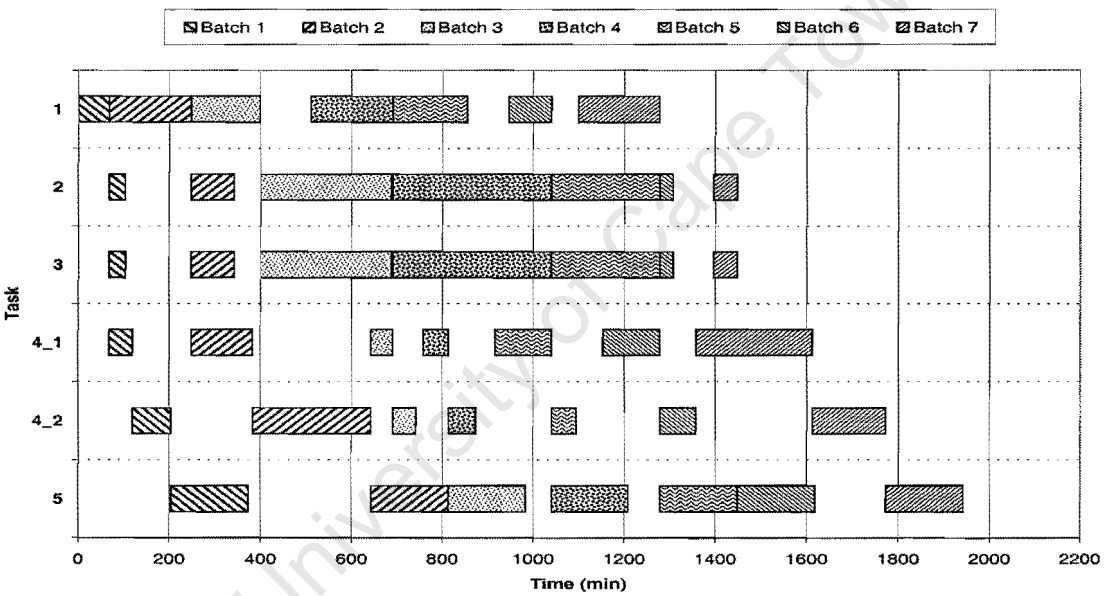


Figure 3.5: Gantt Chart - Change in the sub-division of lots

Scenario 1.3 has a makespan of 1942 minutes, which is an improvement on the 1964 minute makespan obtained in the base case scenario (Scenario 1.1). This is not as big an improvement as was obtained in Scenario 1.2. However, it is not possible to conclude that the order of processing exhibits a more significant impact on the makespan than does the manner in which sources are divided, since none of these scenarios has been optimized. In comparing Figure 3.5 to the base case scenario (Figure 3.3) it is possible to see how the makespan is improved, albeit marginally. The main area where time is recovered is in the processing of the first batch. In

the base case scenario the first batch is more than double the size of the batch in Scenario 2, this causes substantial delays in the initiation of the tasks 2, 3, 4_1 and 4_2 in the second batch, since these tasks cannot finish until task 5 for the first batch has been initiated. From the results of this scenario it can be concluded that the lot size is an important variable in the minimization of the makespan.

3.3.1.4 Scenario 1.4: Sequencing of Tasks 4_1 and 4_2

The fourth and final spreadsheet scenario switches the order of processing tasks 4_1 and 4_2, so that task 4_2 precedes task 4_1 for all batches. This scenario aims to show that the order in which these tasks are sequenced has a definite impact on the makespan. The lot size and order are the same as that used in the base case scenario.

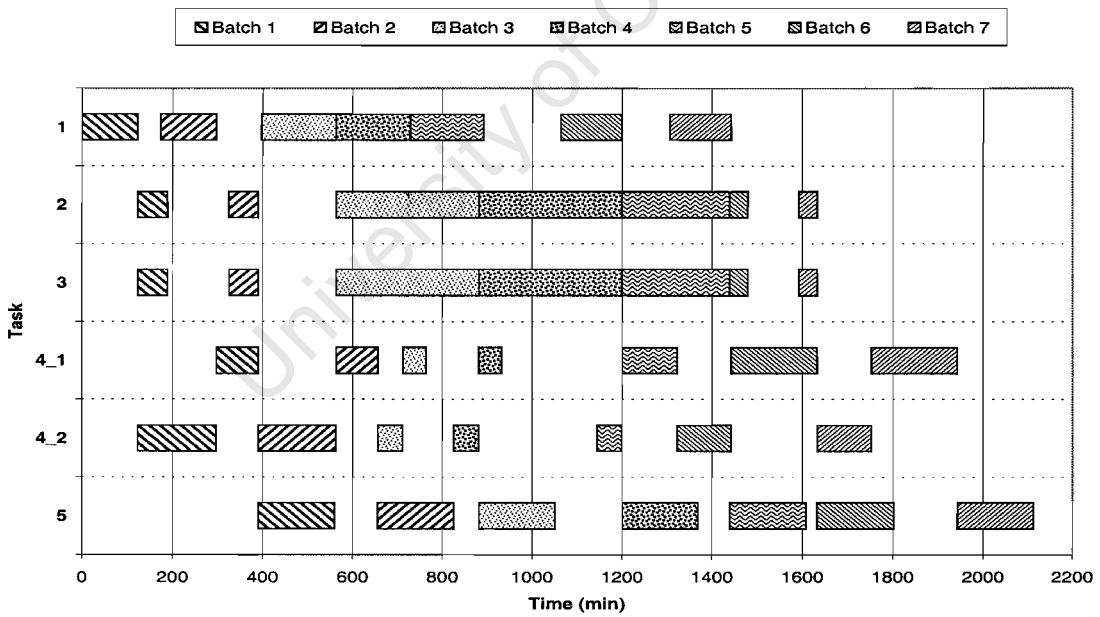


Figure 3.6: Gantt Chart - Switching the order of precedence of tasks on unit 4

The Gantt chart is included here to clearly illustrate how this scenario differs from the base case. The makespan obtained for this scenario was 2113 minutes. If the

start and finish times of tasks 4_1 and 4_2 did not impact on the initiation of other tasks, the order in which these task were processed would be arbitrary. However, as can be seen by comparing the makespan in the base case to that obtained in this scenario, the order in which tasks 4_1 and 4_2 are sequenced can have a large effect on the resultant makespan.

3.3.1.5 Concluding Remarks

From the above scenarios it can be concluded that there is scope for optimization with regard to the determination of the sequencing of lots, the proportions in which material from each source is sub-divided into lots and the sequence in which tasks 4_1 and 4_2 are scheduled.

3.3.2 Scenarios for Optimal Scheduling

As mentioned previously, the problem was formulated into a MILP problem based on the formulation proposed by Ierapetritou and Floudas (1998a). The only economic benefit that may be realized in this plant is in reducing the number of weekly shifts. Therefore the object of all scheduling work is to minimize the makespan. In addition to the completion times of the tasks on unit 5, a penalty term ($\sum_{i,k,n} Pwv_{ikn}$) was included in the objective function. The inclusion of this penalty term is suggested by Ierapetritou and Floudas (1998a) and was discussed in Section 3.2.5 (on page 56). For the scenario studies of this section, a value of one was assigned to the penalty parameter P. The impact of this term on the solution is evaluated later in this section in Scenario 2.9.

Although a large number of scenarios were carried out, only a small number of these scenarios have been included in this section in order to illustrate both the functionality of the formulation and the re-arrangement of the schedule when plant conditions are changed. The mathematical description was set up in GAMS (version 2.50E), which implements the MILP and LP solver, CPLEX (version 6.6). The optimization routines were run on a PII Xeon 550 MHz processor, with 512 Mb of RAM.

3.3.2.1 Scenario 2.1: Base Case

The input conditions to the plant are the same as for the base case in Section 3.3.1.1, except that the order and division of lots are variables. The makespan obtained via optimal scheduling was 1780 minutes. This shows an improvement of between 66 minutes and 333 minutes compared to the arbitrary scheduling of the spreadsheet scenarios. Consideration should be given to the fact that the solution time was quite long, taking 7717 CPU seconds (over 2 hours) and requiring 316033 iterations. The problem consisted of 5098 variables of which 554 were binary variables and was solved to within an integrality gap of 1%. This solution time was obtained with the inclusion of all computational improvements as were discussed in Section 3.2.5.

Table 3.8 summarizes the results obtained, reporting the order of processing and the respective size of each lot, as determined from optimal scheduling.

Table 3.8: Results obtained via optimal scheduling for the base case Scenario

Order	1	2	3	4	5	6	7	Total
Source	1	4	3	2	4	2	1	
Mass	15.0	32.7	45.0	50.0	40.3	41.0	50.0	274.0

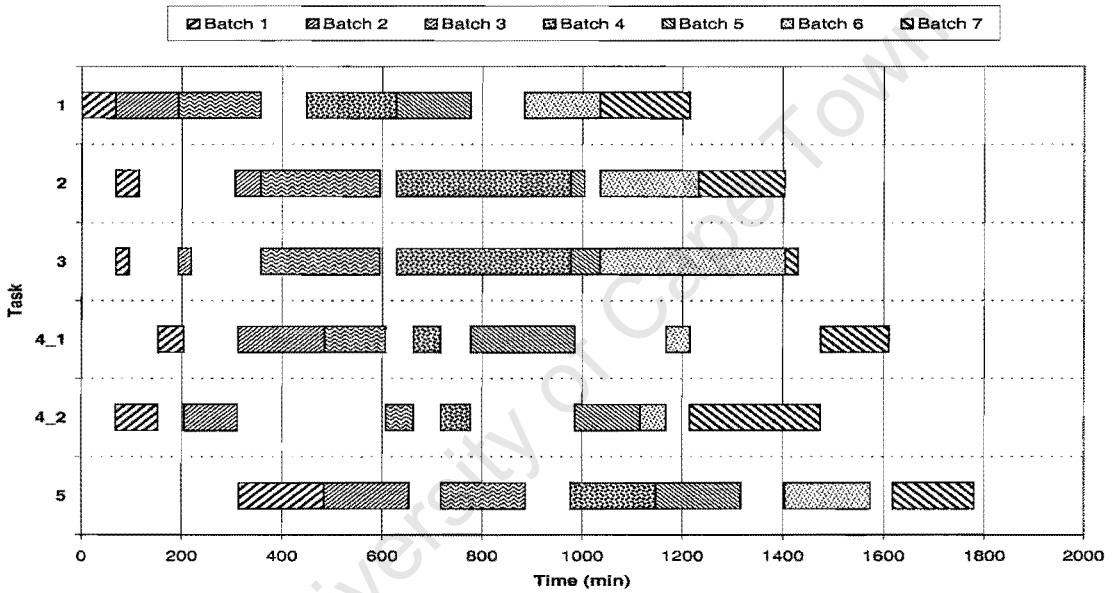


Figure 3.7: Gantt Chart - Optimal scheduling of the base case scenario

Figure 3.7 depicts the optimal schedule in the form of a Gantt chart. Comparing this schedule to those obtained in the spreadsheet scenario it can be seen that the order of processing of the tasks, 4_1 and 4_2 on unit 4 is manipulated so as to reduce the idle times on units 1 and 5. It should be noted that no constraint was included that forces the proportionate division of material between tasks 2 and 3, so as to ensure equal processing times. Therefore in cases where tasks 2 and 3 are not the bottleneck of the system, the division of the material is arbitrary. Whereas in the spreadsheet scenarios the splitting of material between tasks 2 and 3 was determined such that the initiation and completion of these two tasks coincided.

Furthermore, since tasks 2 and 3 produce a large bottleneck when processing material from sources 2 and 3, the reorganization of the schedule puts material from these sources in the most favourable order so as to minimize the impact of this bottleneck. This is a rather intuitive result whereby the scheduling of material from sources which complement each other is alternated. The term 'complement' is used to refer to the fact that the alternate processing of material from some sources is preferred over other possible combinations. This is a direct result of the compositions of each source (see Table 3.3 on page 44) and is based on the fact that one will cause a bottleneck at a particular set of units whereas the other source will cause a bottleneck elsewhere in the plant. Conversely, material which will cause bottlenecking on the same units cannot be said to be complementary.

For example, material from source 1 causes bottlenecking at unit 4 while the material from source 2 causes bottlenecking at units 2 and 3. Therefore it would be preferable to schedule lots from 1 and 2 alternatively as opposed to scheduling multiple lots of material from source 1 followed by multiple lots from 2. Although this may seem to provide a recipe which enables one to predict the order of processing, it is somewhat more complicated because of the fact that there are more than 2 sources of material involved. In the proposed problem there are multiple sources all with different compositions. A situation arises where material from one source may be best complemented with material from another source. However, the converse is not necessarily true as that source may itself be better matched with material from yet a different source. Furthermore, when considered in isolation, two sources may complement each other better than any other combination. However, the system may suffer due to the negative effect that other pairings may cause. Subsequently the combined effect of all complementary pairs must be taken into account.

Moreover, the size of each lot affects the degree of bottlenecking at each unit. Hence this variable can also be manipulated in order to decrease the impact in situations where lots cannot be alternatively scheduled with a complementary source. Evidence of this is shown in the results presented in Table 3.8, by looking at the sequencing of lots from source 4. The first lot from this source has a mass of 32.7 kg and is processed second, following a lot from source 1. The second lot from source 4 has a mass of 40.3 kg and is processed fifth in the sequence between two

lots from source 2. The alternate sequencing of lots from sources 4 and 2 is preferred since these lots complement each other. Hence it is expected that the second lot from source 4 would be larger than the first, due to its more favourable positioning in the schedule.

The processing of material from 4, directly after a lot from source 1 is a somewhat counter-intuitive result since these sources have fairly similar compositions. In fact, swapping the order of processing of the second two lots would appear to provide a more favourable situation, since material from sources 1 and 4 are better complemented with material from source 3 than they are with each other. Thus one would expect the order of processing to be 1-3-4-2-4-2-1 as opposed to that shown in Table 3.8. However, there are a number of reasons why this is not the case.

- Firstly, when processing of the first lot on the first unit begins, a small batch is preferred since all downstream units are initially idle and thus the first task acts as a bottleneck. This phenomenon is due to the fact that the processing time of the first task is dependent on the batch size. The initial bottleneck at the first unit is phased out over the batches which follow and hence the size of the subsequent lots increases.
- Secondly, the total amount of material in source 3 is less than the maximum capacity of a single batch. Although it is possible to sub-divide this batch, the dead times associated with the all tasks penalise any unnecessary sub-division.

For the above mentioned reasons, the preferred scheduling of lot 3 is third in the sequence. If it is to be processed as a single lot, placing it second would cause the downstream units in the plant to be unnecessarily idle at the start of the schedule.

Figure 3.8 presents the cumulative idle times of the units during the processing horizon for Scenario 1.2 of the spreadsheet scenarios and that obtained using optimal scheduling, namely Scenario 2.1. Scenario 1.2 was chosen as a means of comparison since this scenario yielded the smallest makespan in the spreadsheet study.

From the comparative results shown in Figure 3.8, it can be seen that the total idle times associated with each unit obtained via optimal scheduling are much smaller.

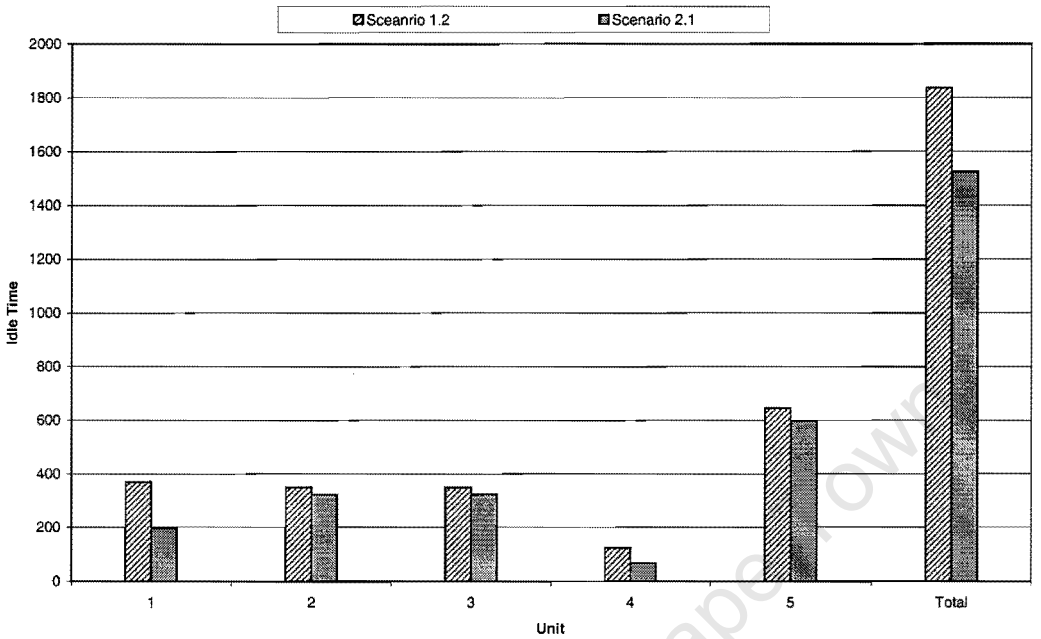


Figure 3.8: Idle times of units in Scenario 1.2 and Scenario 2.1

The bottlenecking on units 2 and 3 is reduced by alternating the scheduling of lots which cause large bottlenecks on these units and those that cause bottlenecking elsewhere. Moreover, the flexibility associated with the division of the material in each source into sub-lots, allows a further reduction in the bottlenecking and ultimately the makespan. The smaller idle times in Scenario 2.1 can be also attributed to the fact that the scheduling of a smaller sized batches initially, reduces the initial idle time on all units.

3.3.2.2 Scenario 2.2: Increase the number of Event Points

The number of event points used in the base case scenario limited the total number of lots to 7, which is the minimum amount of lots that are able to process all the material when considering capacity limitations. This was done for computational reasons, since the addition of each event point introduces a number of binary variables into the problem. The binary variables are indexed over all the event points,

therefore the number of binary variables introduced with each event point is equal to the number of distinct tasks required for each lot. In the plant under investigation there are 32 (4 sources $\times$ 8 tasks) distinct tasks in total and therefore the inclusion of each additional event point increases the number of integer variables in the formulation by 32. Furthermore, the processing of a lot through the plant requires 4 event points, two of which may overlap with the previous lot. For example, the processing of task 2 at a particular event point batch may coincide with the processing of task 5 of the downstream batch. This 'overlapping' does not restrict the final solution in any manner, since it does not result in any resource conflicts due to the time independence of the event points, as was explained in Section 3.2.5 on page 53. Therefore 2 event points must be added to allow the processing of an extra lot, resulting in the addition of a further 64 integer variables to the formulation.

The aim of this scenario is to determine whether the introduction of an extra lot improves the solution obtained in Scenario 2.1. Since the problem would be infeasible if less than 7 lots were processed, it is only necessary to check that increasing the number of lots does not yield a better solution, thus ensuring that the solution obtained in Scenario 2.1 is indeed optimal.

Instead of simply adding the extra number of event points and allowing the optimization routine to determine the optimal number of lots, it was specified that 8 lots must be processed. However, the manner in which material is divided up into lots between the sources was left as a variable. The reason for specifying the number of lots is to reduce the computational effort required in obtaining the solution. To further validify the optimality of Scenario 2.1 another scenario was completed specifying the use of 9 undesignated lots.

Table 3.9: Results obtained specifying the use of 8 lots

Order	1	2	3	4	5	6	7	8	Total
Source	2	1	2	1	4	2	4	3	
Mass	10.0	22.7	31.0	42.2	30.7	50.0	42.3	45.0	274.0

The following results were obtained for the case where the use 8 lots was specified. Material from source 2 constituted 3 lots, sources 1 and 4 were sub-divided into 2

lots each and material from source 3 made up 1 lot. A makespan of 1839 minutes was obtained in this scenario which is greater than that obtained in Scenario 2.1.

Table 3.10: Results obtained specifying the use of 9 lots

Order	1	2	3	4	5	6	7	8	9	Total
Source	1	1	2	4	2	1	3	4	1	
Mass	11.3	14.5	41.2	42.5	49.8	19.6	45.0	30.5	19.6	274.0

The scenario specifying 9 lots had a makespan of 1852 minutes. As was the case in the previous scenario, the number of lots was specified but the sub-division and allocation of material between these lots was left as a variable. In this results of this scenario, source 1 was sub-divided into 4 lots, whereas sources 2 and 4 made up 2 lots each and the material in source 3 was processed as a single lot.

The above results are expected and are a consequence of the associated dead times with the tasks. The dead times tend penalise the use of additional lots and hence the use of the fewest possible lots is optimal in this case.

3.3.2.3 Scenario 2.3: Change in Processing Time

The processing rates of tasks 4_1 and 4_2 used in this scenario were altered from those used in the base case scenario, while the dead times were kept constant (see Table 3.11).

Table 3.11: Processing times of tasks 4_1 and 4_2 for Scenario 2.3

	Task 4_1		**Task 4_2**	
Scenario	**Dead Time** $[min]$	**Proc. Rate** $\left[\frac{min}{kg}\right]$	**Dead Time** $[min]$	**Proc. Rate** $\left[\frac{min}{kg}\right]$
Scenario 2.1	10.0	8.0	10.0	10.0
Scenario 2.3	10.0	8.0	10.0	8.0

Optimal scheduling yielded a makespan of 1645 minutes which is shorter than that determined in the base case scenario (Scenario 2.1). Although this in itself is not an interesting scenario, it does serve to show the importance of accurate plant data.

To further illustrate this point, the makespan was determined using the processing times as specified in this scenario (Table 3.11) together with the sequencing and lot sizing determined in the base case scenario. This yielded a makespan of 1680 minutes which is greater than the makespan obtained using optimal scheduling. This scenario emphasizes the need for accurate plant data when determining the schedule, since the schedule obtained will not necessarily determine the true optimal solution if the data does not accurately reflect the plant.

3.3.2.4 Scenario 2.4: Change in Raw Material Composition

The motivation for the inclusion of this scenario is similar to that of Scenario 2.3, i.e. to emphasize the importance of accurate plant data. In this scenario study the composition of the material in source 4 is altered, the details of which are reported in Table 3.12.

Table 3.12: Fractional composition of source material for Scenario 2.4

Material in Source 4			
	$\mathbf{F_1}$	$\mathbf{F_2}$	$\mathbf{F_3}$
Scenario 2.1	0.10	0.60	0.30
Scenario 2.4	0.30	0.50	0.20

A makespan of 1721 minutes was determined in this scenario which is again shorter than that determined in the base case scenario. As a comparison, the makespan was determined using the compositions obtained from this scenario together with the sequencing and lot sizing determined in the base case scenario. The resulting makespan was 1740 minutes which is worse than the makespan obtained for this scenario.

Due to the fact that the processing times for majority of units in the plant are dependent on the batch size, inaccurate composition data of the material will have a large impact on the resulting schedule. This is a result of the fact that the bottleneck on unit 4 associated with the processing of material from source 4 will decrease, consequently units 2 and 3 must process a larger load. Another point

is that by changing the composition of a particular source, the degree in which material from different sources originally complemented each other is altered.

The results from scenarios 2.3 and 2.4 emphasize the need for accurate plant data, since inaccurate data may result in a sub-optimal solution for the true case. This however, does raise the the question of variability in the system and highlights the disadvantages of controlling a plant where the schedule is determined off-line.

3.3.2.5 Scenario 2.5: Amount of Material in each Source

In this scenario the relative amounts of material in each source was altered, although the total amount of material in the system was kept constant. This scenario is included to show that the relative amount of material in each source has a definite influence on the makespan. The amount of material in each source for this scenario is compared to the amounts used in Scenario 2.1 in Table 3.13.

Table 3.13: Initial mass of material in each source for Scenario 2.5

Source Origin	**Scenario 2.1 Total Mass** $[kg]$	**Scenario 2.5 Total Mass** $[kg]$
1	65	73
2	91	65
3	45	91
4	73	45
Total	274	274

A makespan of 1756 minutes was obtained in this scenario. This improvement on the 1780 minute makespan of Scenario 2.1 is due mainly to the reduction in the amount of material in source 2. Source 2 has a very unbalanced composition and subsequently causes a significant bottleneck in units 2 and 3. Thus the reduction in the amount of material alleviated the impact of this bottleneck and hence the makespan is smaller than that obtained for Scenario 2.1.

3.3.2.6 Scenario 2.6: Demand Constraints

Hard Constraints In the face of customer demands it is often the case that a certain amount of product must be produced by a designated time, regardless of the inefficiency that this may cause in the schedule. If this is the case it is possible to specify demand constraints which are posed as hard constraints, thus ensuring that the product demand is met (assuming it is a feasible demand). To handle this in the formulation the delivery must be tied to an event point, where it is possible to specify the maximum finishing time of a task and also the minimum amount of product that must be produced in that time. The following constraints (equation (3.14)) are an example of how this constraint may be implemented. Note the alternative notation in the representation of the indices.

$$\begin{aligned} T^f(5,5,1,n_5) &\leq T^{max}(5,5,1,n_5) \\ d(S8,1,n_5) &\geq d^{min}(S8,1,n_5) \end{aligned} \tag{3.14}$$

In this example the production demand has been put on the delivery of a minimum amount of material, (d^{min} kg) from source 1, which must be delivered before time T^{max}. The first two indices of the time variable refer to the task and the unit on which the product is processed, i.e. task 5 on unit 5. The third index of the time variable refers to the source from which the material originates (i.e. source 1) and the last index refers to the event point at which this must occur, namely event point n_5. The index 'S8' of the delivery variable refers to the storage unit from which the product must be obtained. In this case the schedule is constrained to the processing of at least d^{min} kg of material from source 1 before any other lot is processed. This is due to the fact that the only one lot may be processed in the first 5 event points.

The fact that the demand constraints have to be related to an event point is somewhat limiting since some knowledge of the schedule is required *a priori*. The particular event point with which the demand constraint is related, adds an additional constraint on the order in which the lot may be processed.

This complication arises in the formulation due to the constraints that ensure the starting time of all tasks that may occur on a unit, is greater than the sum of the processing times of all preceding tasks that occur on that unit (see equation (2.28) on page 29). Thus with every task that is processed on a unit, the calculated start times for all other tasks which may be processed on that unit are also increased. Equation (2.22) requires the finish time of a task to be at least as great as the start time, even if the task does not occur.

Thus if task 'A' occurs at event point 'N_1' followed by task 'B' on the same unit at event point 'N_3', the calculated start time (and hence finish time) of task 'A' at event point 'N_4' is increased to coincide with the finish time of task 'B'. Since task 'A' is not actually processed at this event point, the calculated start time of task 'A' is referred to as its 'superficial' start time. If a demand specification were to be placed on the delivery of an amount of material from task 'A' at event point 'N_4', the superficial finish time of this task at this event point may exceed the imposed time limit, T^{max}, even though the task was actually processed within this limit. The consequence of this happening is that the problem would be infeasible, when in actual fact the demand may well be feasible. Note, this infeasibility only occurs where tasks are assigned to event points. In cases where there is no assignment of tasks to event points, the optimization process would be able to assign the task 'B' to an event point outside the range (*i.e.* $\geq N_5$) of the imposed demand in order to obtain a feasible solution. This would not change the actual solution, since the event points are independent of time but it would require the use of more event points in the formulation, thereby increasing the size of the problem. An alternative is to pose the demand constraints as soft constraints, an issue which is dealt with in the following section.

A scenario was carried out using the base case conditions and specifying the following product demands.

$$\begin{aligned} T_f\,(5,5,1,n_6) &\leq 400\,min \\ d\,(S8,1,n_7) &\geq 15\,kg \end{aligned} \tag{3.15}$$

$$\begin{aligned} T_f\,(5,5,4,n_8) &\leq 560\,min \\ d\,(S8,4,n_9) &\geq 20\,kg \end{aligned} \tag{3.16}$$

In equation (3.15), 15 kg of material from source 1 must be produced with 400 minutes after processing begins. Furthermore this material must be processed within the first 2 lots, this is a consequence of the fact that only 2 lots can be processed in the first 6 event points. The delivery is made before the 7^{th} event point, i.e. the event point after which processing on the last unit is complete. Equation (3.16) specifies that 20 kg of material from source 4 must be delivered within 560 minutes. This material must be processed within the first 3 lots, which is specified by the use of 8 event points.

The results of this scenario are shown in Figure 3.9. A makespan of 1805 minutes was obtained. As expected the overall schedule is worse off than when no delivery constraints are posed (*viz.* Scenario 2.1). In the first lot, 15 kg of material from source 1 was processed, which was available after 374 minutes. The material from source 4 was the second lot to be processed and was available within a time of 555 minutes. In order for the 20 kg of material to be delivered within the time restriction, the lot size can be no larger than 20 kg and hence source 4 must be divided into 3 lots in order to process all material. This is a consequence of the maximum capacity of the units which is 50 kg. Therefore, a total of 8 lots are required in the schedule.

Soft Constraints An alternate situation allows for the demand constraints to be posed as soft constraints thereby allowing late delivery but penalizing the schedule in relation to the tardiness of the delivery.

$$T^f_{ijkn} \leq T^{max}_{ijkn} + P \tag{3.17}$$

where:

P is a the penalty term and is included in the objective function where it will be minimized.

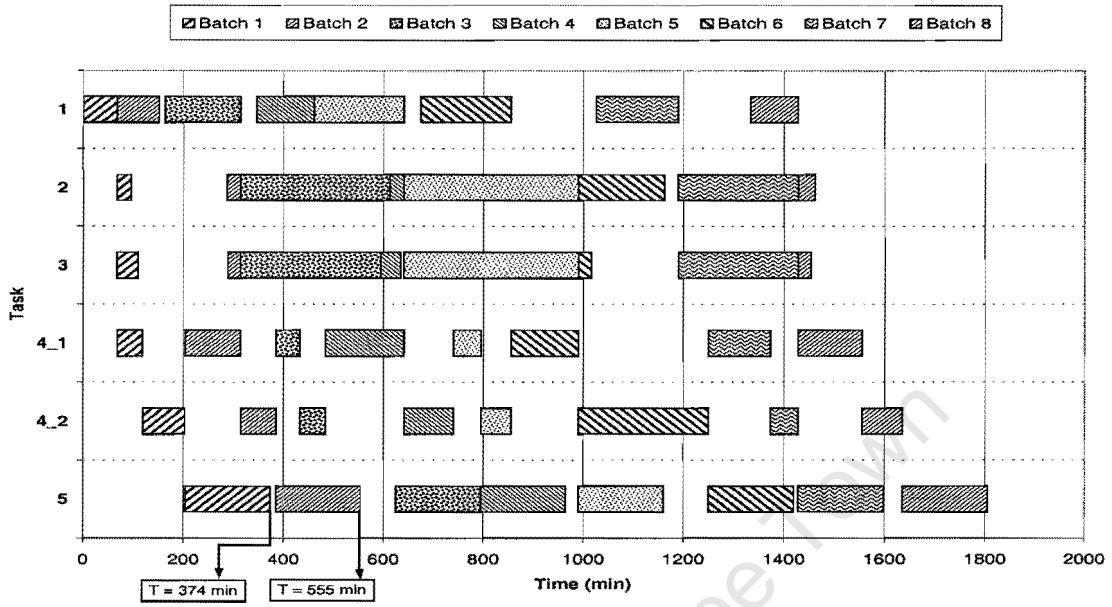


Figure 3.9: Gantt Chart - Scheduling in the face of hard demand constraints

A benefit of using soft constraints is that the solution will not become infeasible in the case where it is not possible for a task to occur within the given time period, as is the case where hard constraints are used. Furthermore, by using soft constraints it may be possible to obtain a smaller overall makespan at the cost of making a slightly late delivery. The weighting of the penalty factor P in the objective function can be manipulated in order to change the relative importance of the overall makespan and the degree of lateness that the product may be delivered. By heavily weighting the penalty factor, it is possible to effectively force the soft constraints to become hard constraints without the possibility of the problem becoming infeasible.

A further scenario was carried out specifying the demand constraints as soft constraints. This is done by including the terms P_1 and P_2 in equations (3.18) and (3.19), respectively. P_1 and P_2 are then minimized in the objective function.

$$\begin{aligned} T_f(5,5,1,n_6) &\leq (400+P_1)\ min \\ d(S8,1,n_7) &\geq 15\,kg \end{aligned} \tag{3.18}$$

$$\begin{aligned} T_f(5,5,4,n_8) &\leq (560+P_2)\ min \\ d(S8,4,n_9) &\geq 20\,kg \end{aligned} \tag{3.19}$$

By weighting the P_i terms in the objective function the following results were obtained. The overall makespan was 1780 minutes, 25 minutes less than that obtained with the demand constraints posed as hard constraints. Material from source 1 was the first lot to be processed in the sequence and had a mass of 15 kg. Material from source 4 was the second lot to be processed. The lot was 25 kg but was delivered 34 minutes late. Allowing the extra 5 kg to be processed with this lot, removes the need to process source 4 as 3 separate lots. Weighing up the difference between the results obtained using hard and soft constraints it is hard to justify tardy deliveries when the overall difference in the makespan is marginal. However, it does indicate that there is a degree of flexibility in schedules which are constrained by demand requirements. Figure 3.10 presents the results obtained in the form of a Gantt chart.

3.3.2.7 Scenario 2.7: Removal of Parallel Unit

A situation which is likely to occur is the maintenance of a particular unit during the processing horizon. For the majority of the units in the plant, the production would have to be halted and the optimal schedule will not change. However, in the case where one of the parallel units (either unit 2 or 3) is removed from service for a period of time the plant may continue operation, albeit at a slower rate. In this scenario, unit 3 is removed from service for a period of 13 hours during the time horizon. The implementation of this condition in the formulation was done by setting the maximum operating capacity for unit 3 to zero, over the designated

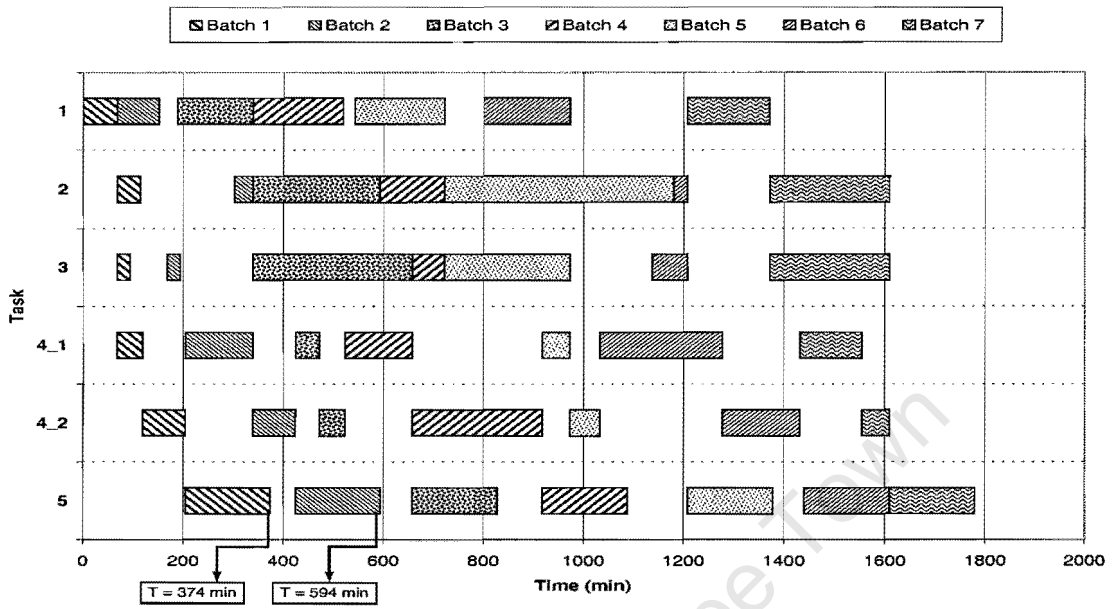


Figure 3.10: Gantt Chart - Scheduling with soft demand constraints

time of unavailability. Table 3.14 reports the ordering and relative size of the lots and Figure 3.11 is a graphical representation of the schedule.

Table 3.14: Scheduling results obtained for Scenario 2.7

Order	1	2	3	4	5	6	7	8	Total
Source	4	2	4	1	2	1	2	3	
Mass	25.2	38.1	47.8	20.7	13.1	44.3	39.8	45.0	274.0

As expected, the bottleneck is emphasized at the point where the unit is removed from service. From the data in Table 3.14, it is seen that material from sources which induce bottlenecking on the parallel units (units 2 and 3) are scheduled outside the period of unavailability of unit 3. More specifically, unit 3 is unavailable for the processing of the 3^{rd}, 4^{th} and 5^{th} lots. The compositions of the source material determine where the material will cause a bottleneck in the plant. Sources 1 and 4 cause bottlenecking on unit 4, while sources 2 and 3 cause bottlenecking on units 2 and 3. Thus the material from sources 1 and 4 constitutes the majority of the material that is processed during this interval of unavailability, in order to reduce the degree of bottlenecking on unit 2. Another interesting point is the fact that the

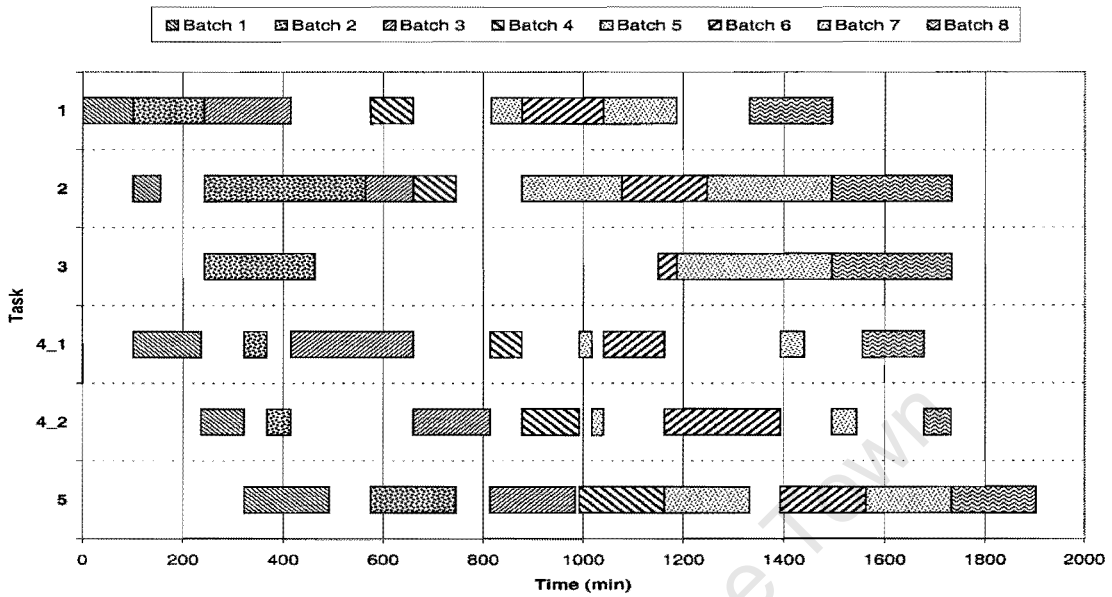


Figure 3.11: Gantt Chart - Removal of a unit from service

use of 8 lots was determined to be optimal. This is to allow a small lot, which does not cause a large bottleneck, of 13.1 kg from source 2 to be processed during the period of unavailability of unit 3.

3.3.2.8 Scenario 2.8: Pre-determined Order

By specifying the order in which lots must be processed, the solution time is reduced to a couple of seconds. The reason why the computational time is reduced so significantly is that the majority of binary variables are in effect removed from the problem, essentially leaving a LP problem. The only discrete decision that must be made is the intra-lot sequencing of material on unit 4.

The inclusion of this scenario may seem illogical since a large degree of flexibility is removed from the problem and the solution will in all likeliness be sub-optimal. However, it is sometimes the case that product orders are pre-determined due to operational constraints and management decisions. Therefore an optimization routine which obtains a sub-optimal solution in a matters of seconds may prove to be more

valuable than the optimal solution which would only be available after a number of hours. Furthermore, once the full schedule has been determined it is possible to make changes to the schedule in real-time if the actual schedule starts to deviate from the off-line schedule. These changes are however limited to the manipulation of the lot size.

The ordering of the lots in this scenario was obtained from the solution of Scenario 2.1. As expected the individual lot sizes and the makespan were the same as those determined in Scenario 2.1. More importantly, the solution time was 15.34 CPU seconds, compared to the 7717 CPU seconds required to obtain a solution in Scenario 2.1.

3.3.2.9 Scenario 2.9: Manipulation of the Penalty Term

Ierapetritou and Floudas (1998b) report that the inclusion of the penalty term, of the form $\sum_{i,k,n}(Pwv_{ikn})$, in the objective function improves the computational performance of the solution (as discussed in Section 3.3.2.2). This performance enhancement is also reported not to affect the optimality of the solution when production demands are posed as hard constraints. Therefore the aim of this scenario is to try and quantify the improvement obtained with the inclusion of this penalty term. In the scenarios carried out thus far, a penalty term was included in the objective function. The value of P in all cases was 1.

To determine the effect of this penalty term, two further scenarios were carried out. In the first scenario (Scenario 2.9i) the value of P in the penalty term was set to 0, thereby removing this term from the objective function equation. In the second scenario (Scenario 2.9ii) the penalty term was set to 20. The following solutions were all solved to within an integrality gap of 4%. The computational results are compared to the base case scenario ($P = 1$) in Table 3.15.

From the results reported in Table 3.15 it appears that the inclusion of the penalty term seems to impact negatively on the solution time. The general trend is that as the weighting is increased so does the number of iterations and subsequently the solution times. This is contrary to that reported by Ierapetritou and Floudas

Table 3.15: Comparisons of the computational effort required

Scenario	Value of P	CPU seconds	Iterations	Makespan
2.1	1	4430	209 097	1780
2.9i	0	4054	156 790	1780
2.9ii	20	5323	283 786	1787

(1998b), who state that the inclusion of the penalty terms aids the solution process. Moreover, the solution obtained in the scenario where a weighting of 20 was used (Scenario 2.9ii), is sub-optimal. This is a consequence of the fact that the solutions were only solved to within an integrality gap of 4%. The additional weighting adds to the magnitude of the objective function and hence the time terms become less significant in the objective function. It should be noted that this is not a conclusive analysis, but it does indicate that the inclusion of the above discussed penalty term does not aid the solution process in all cases.

3.3.3 Computational Issues Associated with Scenarios

3.3.3.1 Computational Time for Solution

In summary, the above scenarios (excluding Scenario 2.8) took between 1 and 7 hours to solve. This is a very large range, the difference between the shortest and longest solution times being nearly an order of magnitude. The very long solution times were experienced with problems which had substantially more binary variables and therefore are expected. However, it was found that there was a significant difference in the processing times obtained for problems of the same size. For example, scenarios 2.1 and 2.4 have the same number of continuous and binary variables. The optimization procedure for Scenario 2.1 took 7717 CPU seconds (2h08m) and required 316033 iterations to solve, whereas Scenario 2.4 solved in 5432 CPU seconds (1h30m) and 199298 iterations. For both of these scenarios the problem was solved to within an integrality gap of 1% (hence the solution times are greater than those reported in Table 3.15). These results indicate that although the number of integer variables provides some indication of the size of the problem,

it is not the only measure of the complexity and size problem. The following issues also play an important role on the computational efficiency of the solution.

- The solver options that are used in the solution of the problem (see Section 3.2.5.3 on page 57). This includes the algorithm specified to solve the relaxed sub-problems, the method in which the candidate variables are selected *etc.*.
- The degree and manner in which the problem is constrained. In general, more constrained LP problems will be more difficult to solve. However, MIP problems benefit from the use of more constraints since the search space is decreased, with the result that there are fewer feasible nodes and the integrality gap is often reduced (Willams, 1993).
- The existence of multiple solutions which have the same optima. In this case the solution procedure often has to continue long after the optimal solution is found, performing an exhaustive tree-search in order to guarantee optimality.

3.3.3.2 Specifying the Number of Lots

In determining the optimal solution it is possible to over-specify the number of lots that may occur and allow the optimization routine to decide the optimal number of lots. However, it was determined that it is preferable, from a computational viewpoint, to specify the number of lots that must occur in the schedule. Starting with the minimum number of lots to ensure a feasible solution, the number of lots is increased until the solutions start to deteriorate. As was the case with all the scenarios (excluding Scenario 2.7) it was determined that the minimum number of lots always proved to be optimal. This is a consequence of the large dead times associated with tasks.

In scenarios where only 7 lots were used, the problem had 554 integer variables. In the case where more than 7 lots were required to complete processing (Scenarios 2.2 and 2.7) the number of integer variables increased to 616 integer variables for 8 lots and 702 integer variables for 9 lots. The solution time increased substantially as a result of the larger number of integer variables associated with these problems.

3.3.3.3 Batch-size Dependent Processing Times

It was suspected that a large degree of the computational effort required in the solution to the problems in the scenario study was due to the fact that the processing times of tasks are dependent on the size of the lots. In order to confirm this suspicion a test scenario was completed in which the lot sizes were made independent of the batch size. With the processing times being independent of the batch size, the task duration no longer depends on the relative compositions of the source material. Therefore to ensure the order in which lots are processed is still an important variable in the minimization of the makespan, tasks from different sources were assigned unique processing times. These processing times were based on the solutions obtained in the scenario study, so as to reflect the original problem. This of course changes the problem to a degree but it does provide some insight into the complications introduced by using batch-size dependent processing times.

The investigation showed that using batch-size dependent processing times does indeed worsen the solution procedure of the problem. In the scenario where the processing times are batch-size independent, the first integer solution was obtained on the 10^{th} node in the tree-search which had a relative integral gap of 9.20%. In the scenario study, where the processing times were batch-size dependent, the first integer solutions were only found after the investigation of more than a hundred nodes. Furthermore, the integrality gap at the first node was in excess of 35% for all these scenarios. As discussed in Section 3.2.5.2 (on page 56), the larger the integrality gap the larger the search time. In addition to this, the location of integer solutions deep in the tree search requires the investigation of a larger number of nodes before a solution can be declared optimal.

The worsening of the solution procedure in the case where processing times are dependent on the batch size is attributed mainly to the increase in the integrality gap. This increase is due to the fact that the processing times are now dependent on the values of both the binary variables (wv) and the batch size (B), thus allowing the relaxed solution more flexibility with which to improve the relaxed objective function. This characteristic is emphasized for the case where the objective is to minimize the makespan, since partial allocation of tasks is encouraged even further

so as to allow processing times and hence the objective function to be kept to a minimum.

The optimal solution, using batch-size independent processing times, was found in 1867 CPU seconds after 28667 iterations. Thus it can be concluded that using batch-size dependent processing times does indeed impact negatively on the solution times of scheduling problems.

Chapter 4

The Integrated Batch and Continuous Plant

This chapter deals with the addition of a continuous process upstream to the batch plant described in Chapter 3. A STN diagram of the plant is shown in Figure 4.1. Sources 1-3 require pre-processing in the continuous plant. The fourth source enters the plant at unit 1, bypassing the continuous process. A difficulty associated with the continuous plant arises from the fact that the plant operates in a nonlinear fashion as a consequence of its efficiency being related to the processing rate. Operating at a low rate has the benefit of decreasing the load on the downstream units although the time taken to process material is longer. If the plant is operated at a fast rate the resulting low separation efficiency causes more material to report to the product stream. In order to handle this nonlinearity a number of approximations can be made. Two different approximations are presented, followed by a scenario study.

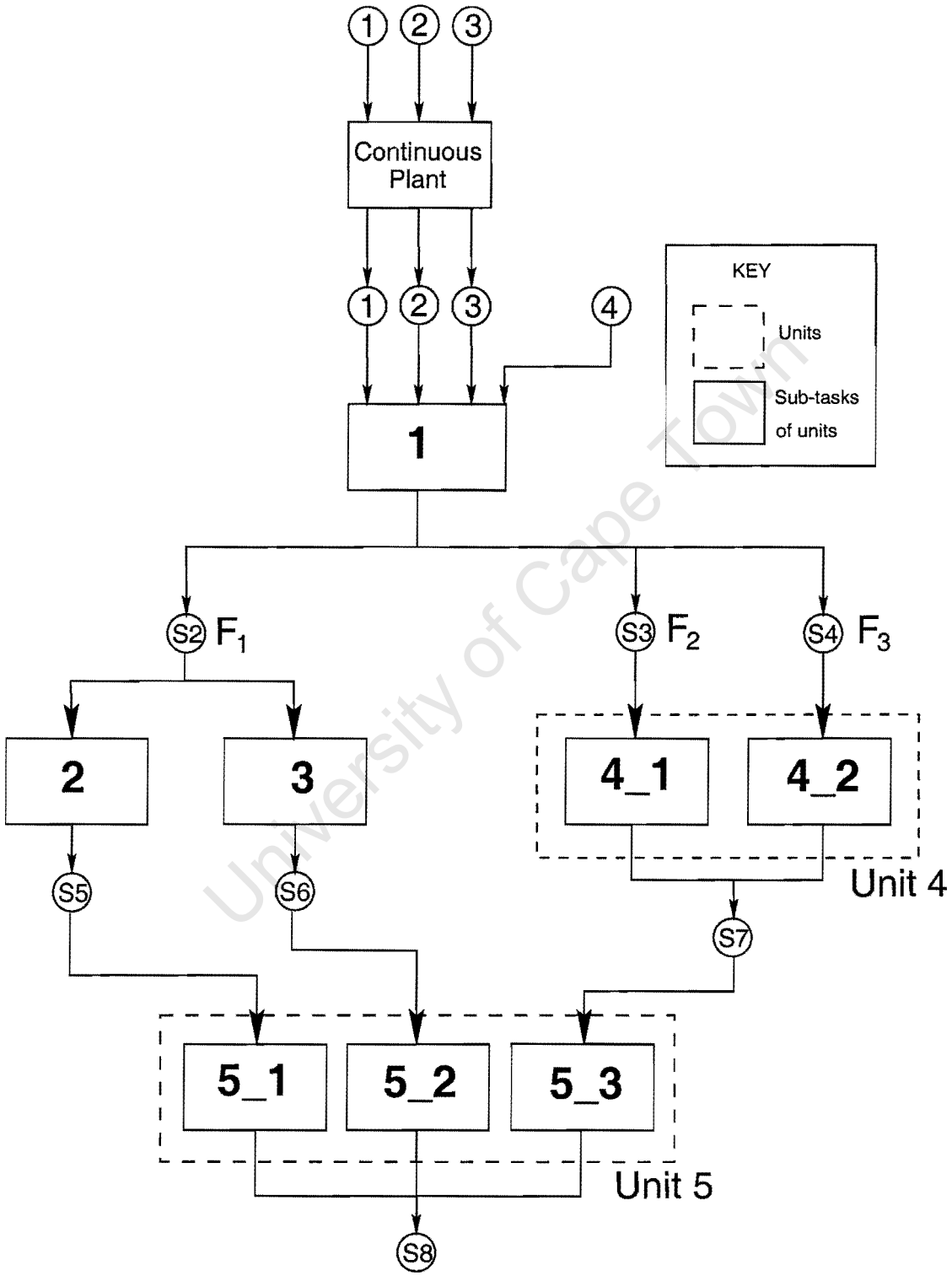


Figure 4.1: Modified STN diagram of the integrated batch and continuous plant

4.1 The Continuous Plant

4.1.1 Description

The continuous plant is comprised of a number of processing units which can be simply modeled as a black box (see Figure 4.2). This approximation is validated by the fact that a particular unit forms the bottleneck in the plant and thus no generality is lost in approximating it as a black box. The plant forms an initial purification stage upstream from the batch plant and has the flexibility to operate over a range of processing rates. The relationship between the processing rate and the mass of material in the concentrate stream is nonlinear, with the general result that as the rate of processing increases the less efficient the operation and thus the larger the mass of concentrate reporting to the downstream batch plant. The plant has the unique feature of ensuring that all valuable material reports to the product stream and thus the processing rate essentially determines the amount of gangue material in the product stream. Due to the continuous nature of the plant, the amount of material produced is also dependent on the time allocated for processing (τ).

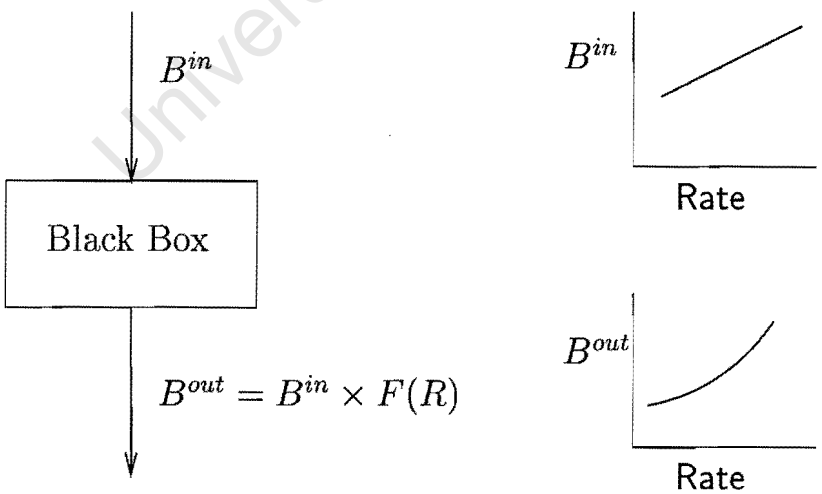


Figure 4.2: Black box approximation of the continuous plant

The 3 different sources are processed through the continuous plant separately *i.e.* no mixing may occur. The order of processing influences the availability of material to

the downstream batch plant and is therefore classed as a variable. The processing of a particular source may be interrupted in order to process another source. However, no mixing of the 3 sources is allowed. It is assumed that altering the processing rate in the continuous plant does not influence the product quality of the material with respect to the relative fractions in which material is divided between units 2, 3 and 4. Hence the values for F_1, F_2 and F_3 remain unchanged. This assumption is valid if the gangue material is distributed between the fractions in the same proportions as the valuable material.

The total amount of material of each of the sources 1-3 that is processed is given in Table 4.1. The total output mass is listed as a range, since it depends on the rate of processing. The smaller values in the range coincide with the minimum processing rate, while the larger values coincide with the maximum processing rate. The material exiting from the continuous plant is deposited into a storage bin which feeds the downstream batch plant. Each source has a designated storage unit, which for practical purposes can be assumed to have no capacity limits. The size of the lots entering the batch plant are restricted by the availability of material from the continuous plant as well as the capacity limitations of the batch plant.

Table 4.1: Input and output ranges for the continuous plant (Sources 1 - 3)

Source	**Total Input Mass** $[kg]$	**Total Output Mass** $[kg]$
1	150	67.5 - 76.5
2	170	76.5 - 86.7
3	60	30.6 - 27.0

In summary, the flexibility of the continuous plant is due to the following variables:

- processing rate
- time allocation for each lot
- order of processing

These variables all interact to determine the amount of material as well as the time made available for processing in the downstream plant.

4.1.2 Mathematical Formulation

For the purpose of scheduling, the continuous plant may be treated as a batch process because the downstream unit is a batch plant and hence cannot take advantage of the continuous production of material. The size of each lot in the continuous plant is dependent on the rate of processing and the time available for processing material. The mass of material reporting to the product stream is determined by multiplying the mass of material that enters the unit by a fraction, $F(R)$. This fraction is related to the rate by means of a linear function, *viz.* equation (4.1). Note that this relationship is only valid for the range $[R^{min}, R^{max}]$ and for $R \neq 0$. In the development of the formulation only the relevant indices are quoted, subsequently the indices i, j, k, n relating tasks, units, source origins and event points, respectively have been omitted.

$$F(R) = bR + c \tag{4.1}$$

where:

b and c are model parameters of the plant, relating the processing rate to the exiting fraction F.
R is the rate of processing $\left[\frac{mass}{time}\right]$.

The mass of material reporting to the product stream (B^{out}) is therefore a function of the processing rate (R) and the amount of material entering the unit (B^{in}). Due to the fact that both R and B^{in} are variables, the relationship used to determine the amount of material reporting to the product stream is nonlinear, *viz.* equation (4.2). The remaining fraction of material reports to the waste stream and is not accounted for since it has no further impact on the process.

$$B^{out} = F(R) \times B^{in} \tag{4.2}$$

where:

B^{in} the mass of material in the feed to the continuous plant.
B^{out} the mass of material reporting to the product stream.

The nonlinearity arises in the calculation of the processing time. This is due to the fact that the processing time is dependent on the mass of material entering the continuous plant and the processing rate, both of which are variables, *viz.* equation (4.3).

$$T^f - T^s = \alpha \times wv + \frac{B^{in}}{R} \tag{4.3}$$

where:

T^f is the finishing time of a task.

T^s is the start time of a task.

α is the dead processing time.

wv is the binary variable which is one if the task is being processed otherwise it is zero.

The batch plant may process any amount of material (within its capacity limits) as long as this material is available in the upstream storage units. Thus the operation of the continuous plant is to a large degree dependent on the operation of the batch plant, since the continuous plant must ensure there is material available for processing in the batch plant.

The processing of tasks on the batch plant requires the use of 4 event points, where the processing of subsequent tasks on Unit 1 may only occur every second event point, which is suffient to prevent any resource-task conflict between different lots (as explained in Section 3.2.5 on page 53). Thus the continuous plant may process 2 tasks for every lot that is processed on the batch plant. This introduces a degree of flexibility into th operation of the continuous plant, since a single task may be processed as 2 separate tasks. By allowing a task to be processed as two separate tasks a portion of the original task may be processed at a slow rate, while the other portion may be processed at a fast rate. Thus the continuous plant can be manipulated such that the amount of material reporting to the batch plant is minimized without causing a delay in initiating the processing on the batch plant.

4.1.2.1 Approximation using Linear Transformation

Approximation of the Rate of Processing

In order to linearize the above relationship it is necessary to discretize the processing rate into a number of values between the maximum and minimum bounds of the processing rate of the unit, *viz.* equation (4.4). This has the implications that the result may be sub-optimal as a consequence of the rate not being able to take on all values between the maximum and minimum processing rate. The notation used in the formulation is to represent indices as subscripts.

$$\frac{1}{R} = \frac{X_1}{a_1} + \frac{X_2}{a_2} + \frac{X_3}{a_3} + ... + \frac{X_m}{a_m} \tag{4.4}$$

where:

a_m are the possible values of the processing rate and $a_m \subset [R^{min}, R^{max}]$.
X_m is a binary variable associated with each value a_m that the rate may take.
m is the number of discrete rates at which the plant may operate.

Equation (4.4) is expressed with the inverse of the rate as the subject of the formula and the binary variables as the numerator of each term. This is done to ensure that when equation (4.4) is substituted back into equation (4.3), the equation does not become undefined for the case where the binary variables take on a value of zero.

In equation (4.4), as $m \rightarrow \infty$ the approximation of the rate tends towards the true situation, where the rate is a continuous variable. However, as m increases so does the number of integer variables in the problem and hence the required tree-search also increases. The tree-search can be as much as doubled with the addition of each additional binary variable. Therefore it is necessary to make a compromise between the degree of accuracy of the solution and the computational time, by limiting the number of integer variables in the approximation of the rate.

To ensure that only one value is selected for the rate out of the possible set of values, the following constraint is implemented in the formulation. This ensures that at most only one of the integer variables, from the subset X_m, at any event point can

be equal to one.

$$\sum_m X_m \leq 1 \tag{4.5}$$

Approximation of the amount of material reporting to the product

The amount of material reporting to the product stream is determined by multiplying the amount of material entering the plant by some fraction (F_m). The value of the fraction is determined by the processing rate and hence the resulting relationship is nonlinear. Using equation (4.1), it is possible to determine the corresponding value of F_m for each of the possible values of R in equation (4.4), as follows.

$$F_m = ba_m + c \tag{4.6}$$

Substituting equation (4.6) into equation (4.2), the following relationship is obtained.

$$\begin{aligned} B^{out} &= B^{in} \sum_m F_m X_m \\ &= F_1 X_1 B^{in} + F_2 X_2 B^{in} + ... + F_m X_m B^{in} \end{aligned} \tag{4.7}$$

where the set of binary variables X_m correspond to those used in equation (4.4). This is to ensure that the correct fraction value F_m is associated with the rate value a_m in equation (4.4).

It should be noted that equation (4.7) is still nonlinear due to the multiplication of a binary variable (X_m) with a continuous variable (B^{in}). However this can be dealt with using the following transformation (Glover, 1975).

$$\text{Let } X_m B^{in} = B_{Xm}$$

$$\Rightarrow \quad B^{out} = F_1 B_{X1} + F_2 B_{X2} + ... + F_n B_{Xm} \tag{4.8}$$

Here B_{Xm} is a continuous variable. Then the following set of constraints ensure the

validity of the transformation.

$$\left.\begin{array}{rcl} B_{Xm} & \leq & B^{in} \\ B_{Xm} & \geq & B^{in} - M\left(1 - X_m\right) \\ B_{Xm} & \leq & MX_m \\ B_{Xm} & \geq & 0 \end{array}\right\} \tag{4.9}$$

where:

M is a sufficiently large number to ensure the above equations are never violated. However, this number should be as small as possible to reduce the integrality gap. Therefore it is suggested that the value for M should be the maximum capacity of the unit (*i.e* $M \geq B^{in}$).

Approximation of the Processing Rate

In equation (4.3), the nonlinearity is a result of the following term, which is represented as τ.

$$\tau = \frac{B^{in}}{R} \tag{4.10}$$

Substituting the discrete approximation for R (equation (4.4)) into equation (4.10) yields the following.

$$\tau = \frac{B^{in}X_1}{a_1} + \frac{B^{in}X_2}{a_2} + \frac{B^{in}X_3}{a_3} + \ldots + \frac{B^{in}X_m}{a_m} \tag{4.11}$$

Now, using the same transformation (equation (4.9)) as before we obtain a linear function, *viz.* equation (4.12).

$$\tau = \frac{B_{X1}}{a_1} + \frac{B_{X2}}{a_2} + \frac{B_{X3}}{a_3} + ... + \frac{B_{Xm}}{a_m} \tag{4.12}$$

Substituting this back into equation (4.3), we obtain the equation used to determine the processing time.

$$\begin{aligned} T^f - T^s &= \alpha \times wv + \frac{B^{in}}{R} \\ &= \alpha \times wv + \left(\frac{B_{X1}}{a_1} + \frac{B_{X2}}{a_2} + \frac{B_{X3}}{a_3} + ... + \frac{B_{Xm}}{a_m} \right) \end{aligned} \tag{4.13}$$

Implementation of the Formulation

Implementing the above formulation into the model, equations (4.8), (4.9) and (4.13) are used. However, additional constraints are also required to maintain the integrity the of the model.

The integer variable X_m is related to the initiation of the task in equation (4.14), where the complete notation is used to represent the indices of the variable X.

$$\sum_m X_{mn} = \sum_k wv_{x_i kn} \qquad \forall \ k \in K, n \in N \tag{4.14}$$

where:

X_{mn} is a binary variable associated with each value a_m at event point n.
x_i is the task representing the continuous plant.

The timing constraints which were presented in Section 2.3.3 (on page 23), also apply to the continuous plant. However, to avoid repetition they are not re-presented here.

4.1.2.2 Alternate Approximation

An alternate approach which can also be used to approximate the nonlinearity is presented in this section. The level of approximation is equivalent to the formulation proposed in Section 4.1.2.1. However, the model is much simpler and takes advantage of the method in which scheduling problems are formulated.

As before the processing rate is divided into a number of discrete points between the maximum and the minimum rate. The exact number of points is dependent on the desired level of accuracy. The continuous task is then approximated as a number of parallel tasks as in Figure 4.3. The task binary variable wv_{x_ikn} is associated with each of the parallel tasks and the equation (4.15) ensures that at most only one of these units may be operational at any one time.

$$\sum_{x_i}\sum_{k} wv_{x_ikn} \leq 1 \qquad \forall\ i \in m, n \in N \tag{4.15}$$

where:

x_i is the i^{th} task in the set of parallel tasks used to approximate the nonlinearity.

m is the set of parallel tasks used in the approximation.

Each of these tasks is then associated with a particular processing rate and a fraction value ($F(x_i)$) to determine the portion of material that exits the process.

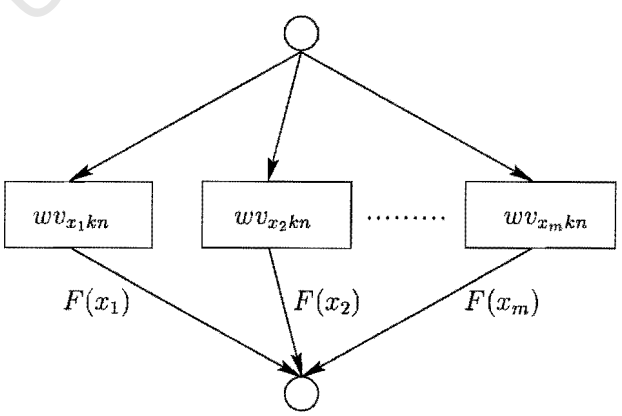


Figure 4.3: Alternative approximation of the continuous plant

4.2 Scenarios

Once again a number of scenarios was carried out to show the functionality of the formulation. The scenario study in Chapter 3 covers many aspects that are also characteristic of the combined continuous-batch plant. Subsequently, the scenario study is not as extensive as that of Chapter 3 and only scenarios which illustrate certain aspects are presented in this section. The following parameters for the continuous plant were used in the scenario study.

$$R^{max} = 1.30 \left[\frac{min}{kg}\right] \qquad R^{min} = 2.80 \left[\frac{min}{kg}\right]$$

$$F(R) = -0.040R + 0.562$$

where:

R^{max} and R^{min} refer to the maximum and minimum processing rates of the continuous plant.

F is the fraction of material entering the continuous plant that reports to the product stream (*viz.* equation (4.1)).

4.2.1 Scenario 3.1: Base Case Scenario

In the base case scenario the nonlinearity was handled using the approximation based on the use of parallel units. In order to keep the solution times to a minimum it was decided to approximate the nonlinearity of the continuous plant using only 2 discrete rates. The problem consisted of 8021 variables of which 654 were binary. The solution required 3359 CPU seconds and 160898 iterations to obtain a solution within an integrality gap of 5%. It is interesting to note that this scenario had a hundred more integer variables than the problem of Scenario 2.1, yet there is little difference in these problems when comparing the computational results. This result can be explained by the fact that MIP problems benefit from the use of more constraints, since the search space is decreased and subsequently there are fewer feasible nodes and the integrality gap is often reduced (Willams, 1993).

The amount of material in each source at the beginning of the horizon is presented in Table 4.1. The rate values and associated product fractions for the scenario are reported in Table 4.2. To keep consistency with the formulation presented in Section 4.1.2, the processing rate is reported as $\left[\frac{min}{kg}\right]$. The continuous plant is modeled as a set of parallel units (x_i), each with an associated processing rate and product fraction. The product fraction is the portion of material entering the continuous plant that reports to the product stream, as determined from equation (4.1).

Table 4.2: Processing rates and product fractions for the continuous plant

Unit	**Processing Rate** $\left[\frac{min}{kg}\right]$	**Product Fraction**
x_1	1.30	0.51
x_2	2.80	0.45

A makespan of 1648 minutes was obtained in the base case scenario, which is less than the optimal makespan obtained in for the batch plant in Scenario 2.1. This is a consequence of the smaller amount of material that the plant processes due to the continuous plant. The optimal schedule is shown graphically in Figure 4.4. The different processing rates for the continuous plant are represented as separate tasks (x_1 and x_2) for clarity.

Figure 4.4 shows that the continuous plant was only operated at the slower rate (x_1) for two periods over the time horizon. As expected material from the fourth source is the first lot to be processed, since it does not require pre-processing in the continuous plant.

Tables 4.3 and 4.4 give a more detailed description of the determined schedule. Comparing the data in these tables it is noticeable that the order of processing in the continuous plant differs from that in the downstream batch plant. Another interesting point is the fact that the initial lot sizes in the continuous plant are relatively small and progressively increase. This is an intuitive result as the continuous plant needs to make material available to the downstream plant as soon as possible at the beginning of the schedule.

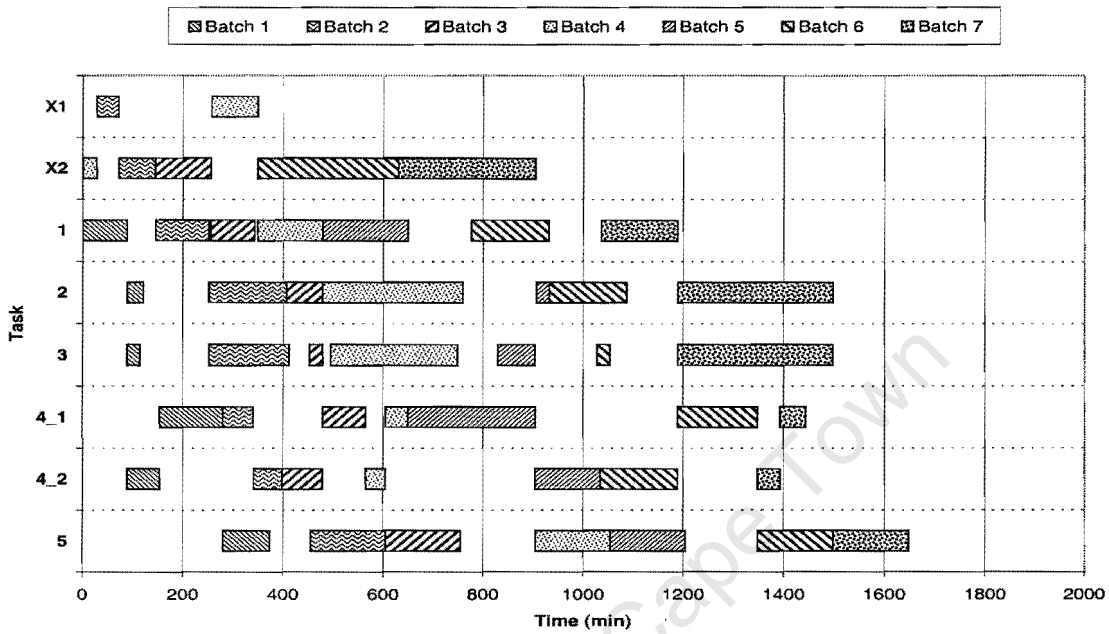


Figure 4.4: Gantt Chart - Scheduling of the integrated batch-continuous plant

Table 4.3: Lot division and order of processing for the material in the continuous section

Order of Processing	1	2	3	4	5	6	7	Total
Source Origin	1	3	3	1	2	1	2	
Rate of Processing	x_2	x_1	x_2	x_2	x_1	x_2	x_2	
Mass of Lot $[kg]$	10.0	34.0	26.0	39.6	71.8	100.4	98.2	380.0

Table 4.4: Lot division and order of processing for the material in the batch section

Order of Processing	1	2	3	4	5	6	7	Total
Source Origin	4	3	1	2	4	1	2	
Mass of Lot $[kg]$	23.0	29.0	22.3	36.6	50.0	45.2	44.2	250.3

4.2.2 Scenario 3.2: Comparison of the Two Approximations

In this section a comparison of the two approximation formulations, presented earlier in the chapter, is made.

Through test work it was determined that the method of approximation presented in Section 4.1.2.2 is preferred because of its simplicity, but more importantly it is superior with respect to the computational efficiency of solution. The first approximation method results in a formulation where the first integer solution has a very large integrality gap. After allowing the optimization procedure to run for more than 72 hours, the relative integrality gap was still in excess of 30%. The makespan at this stage was 1713 minutes, clearly sub-optimal when compared to the makespan obtained in Scenario 3.1. Subsequently, the formulation which was used for the scenario study, was developed and implemented.

4.2.3 Scenario 3.3: Change in the Level of Approximation

In this scenario the level of approximation was increased from 2 discrete rates (Scenario 3.1) to 3. Table 4.5 reports the processing rates and the associated product fractions for the 3 parallel units used in the approximation of the continuous plant. In contrast to the base case scenario, namely Scenario 3.1, the number of variables in this problem was 9617, of which 697 were integer variables. The problem took 716399 iterations and 11037 CPU seconds to solve to within the same integrality gap as the base case scenario.

Table 4.5: Processing rates and product fractions for the continuous plant

Unit	Processing Rate $\left[\frac{min}{kg}\right]$	Product Fraction
x_1	1.30	0.51
x_2	2.05	0.48
x_3	2.80	0.45

Increasing the number of discrete rates in modeling the continuous plant increases the accuracy of the approximation. Subsequently, the flexibility of the model is also

increased. Thus it is expected that the makespan obtained when using 3 discrete rates would be the same, if not better than that obtained using only 2 discrete rates in the approximation. This is indeed the case, where the makespan using 3 discrete rates was determined to be 1598 minutes, a 50 minute improvement on that obtained for the base case. The Gantt chart for this scenario is shown in Figure 4.5.

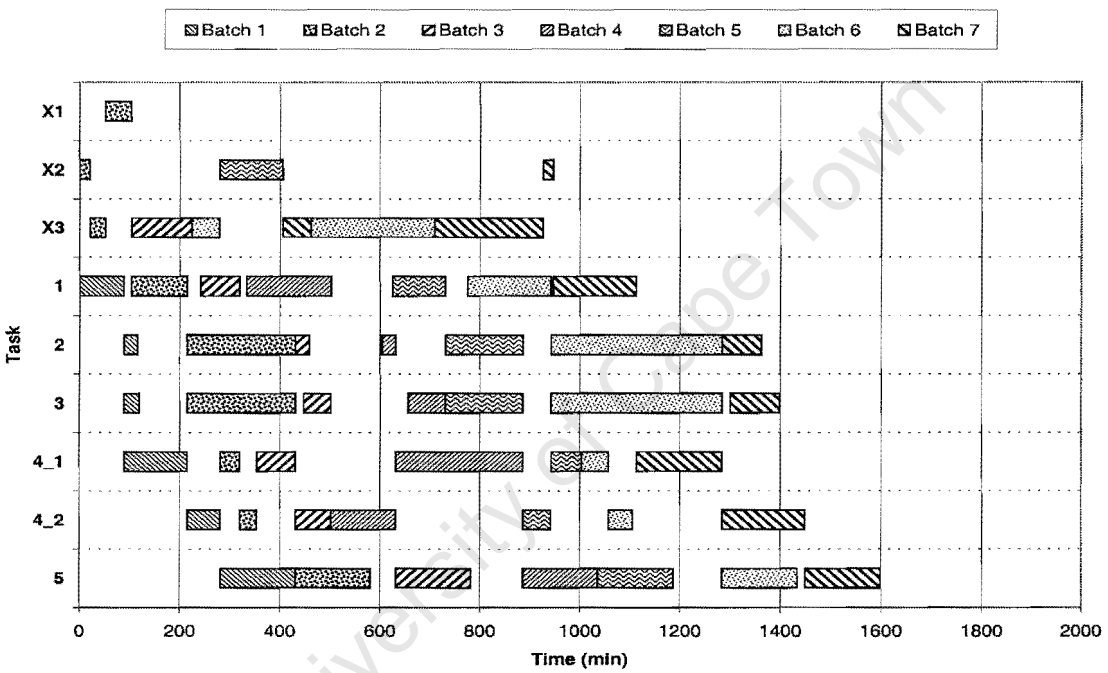


Figure 4.5: Gantt Chart - Increasing the level of approximation of the integrated batch-continuous plant

Figure 4.5 shows that the inclusion of the extra discrete processing rate increases the flexibility of the operation of continuous plant. For example, the last 2 lots processed on the continuous plant are from the same source, but are combined to be processed in the downstream batch plant. The processing of this material is manipulated in the continuous plant, by changing the processing rate, so that the minimum amount of material is transfered to the batch plant without causing a delay in the initiation of processing.

Concluding Remarks

In this chapter two methods for modelling the nonlinearity in the plant were presented. As shown in the scenario study, the two methods are equivalent in terms of the degree with which the nonlinearity is approximated. However, the second method yielded a superior formulation with respect to the computational efficiency of the solution.

Another important aspect addressed in the scenario study, is increasing the number of discrete points in the approximation of the processing rate. In this scenario it was shown that using more discrete point to approximate the rate provides the model with a greater degree of flexibility and hence a better solution was obtained. However, the downside of increasing the number of discrete points is that the solution time is increased significantly.

Chapter 5

Extension to an Industrial Problem

In this chapter, the problem is extended to the full industrial plant. A detailed description of the plant forms the first section of this chapter, after which the formulation of the MILP problem is considered. An alternative method for obtaining a solution is also proposed. This method makes use of a random number generator in order to remove a large number of binary variables from the problem and hence speed the solution time considerably. This method will in all likeliness produce a sub-optimal solution. Thus a compromise may have to be made between the quality of the solution and the time in which it is obtained.

The extensive scenario studies in Chapters 3 and 4, demonstrate the flexibility of the formulation in the modeling of various situations that may arise. Thus the focus of this chapter is on the methods used in obtaining the solution of a more complex industrial problem, as opposed to illustrating the ability of the formulation to handle possible eventualities.

5.1 Plant Description

The industrial plant has a similar construct to the simplified plant described in Chapter 4. However, there are 2 additional units and 6 extra tasks, which result in a more complex flow of material through the plant. Furthermore, products are able to follow different paths through the plant and thus the flowshop classification of the plant changes from a multiproduct to a multipurpose plant.

The amount of material initially available in each source is presented in Table 5.1. Note that the material from sources 1, 2 and 3 are pre-processed in the continuous plant before entering the batch plant, while material from the fourth source is fed directly into the batch plant.

Table 5.1: Initial mass of material in each source

Source Origin	**Total Mass** $[kg]$
1	150
2	170
3	60
4	73

The following description of the plant corresponds to the STN diagram in Figure 5.1. In the work on the simplified plant in Chapters 3 and 4, the processing rates of certain tasks were manipulated to account for the tasks that had been omitted. Thus the processing times of tasks in the industrial problem differ and are presented below, together with a detailed description of the plant.

Continuous Plant

The operation of the continuous plant is as described in Chapter 4, where a detailed description of the nonlinear relationship between the processing rate and the amount of product is presented. The continuous plant is again modeled as a black box and is thus referred to as a single unit. No mixing of material between different sources is allowed in this unit, although the size of lots (effectively, the processing time), the

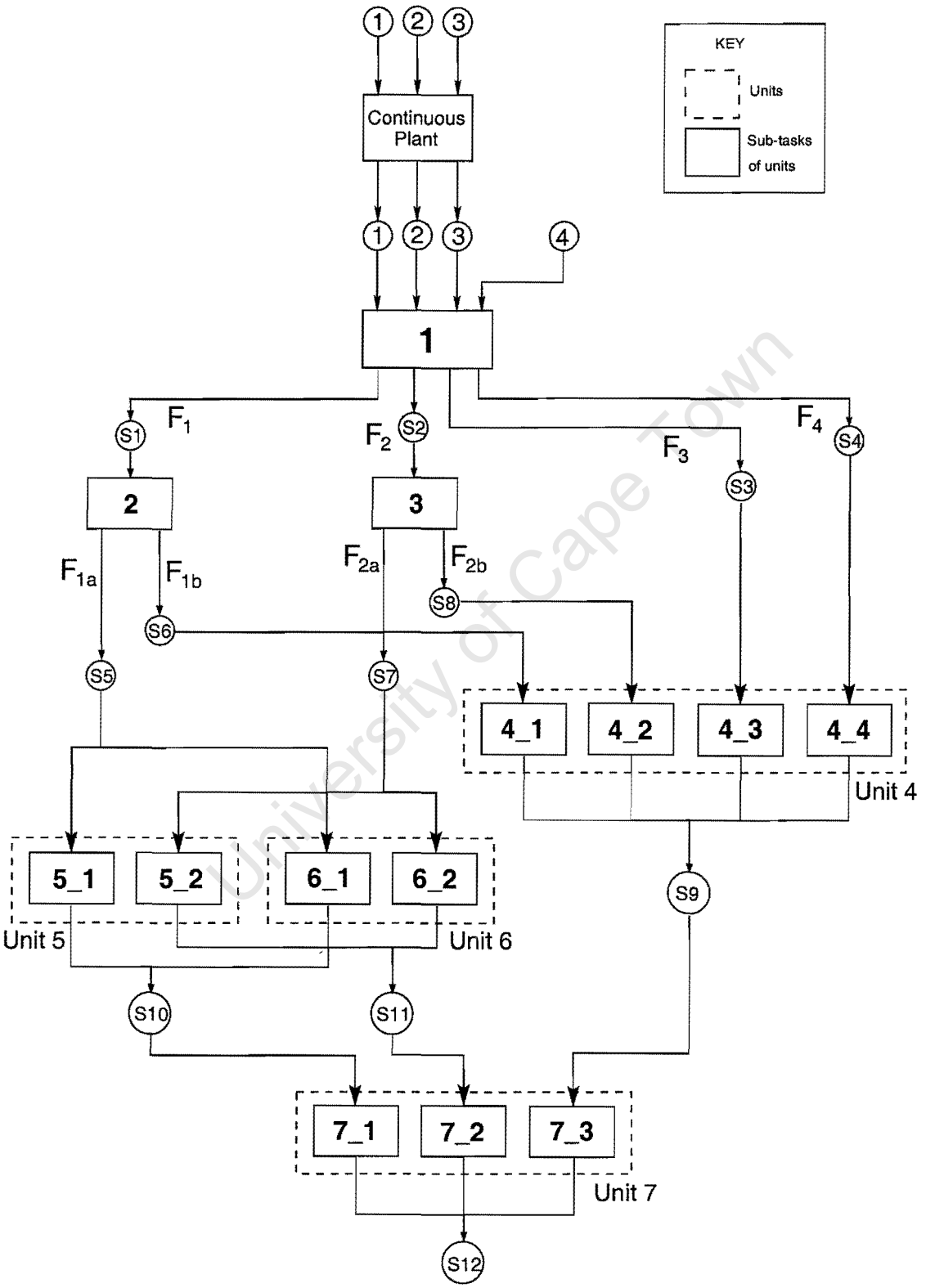


Figure 5.1: State-Task Network of the Full-Scale Industrial Plant

order of processing and the rate at which material is processed are all optimization variables. A unique storage unit is available for the products of each source.

The maximum and minimum processing rates for the unit are presented below, together with the values of the parameters b and c substituted into equation (4.1), shown below as equation (5.1). This equation is used to relate the processing rate to the fraction of material reporting to the product stream.

$$R^{max} = 7.0 \left[\frac{min}{kg}\right] \qquad R^{min} = 2.0 \left[\frac{min}{kg}\right]$$

$$F(R) = -0.14R + 0.538 \tag{5.1}$$

Table 5.2 provides the input and output data as determined from the above data. The range of the output mass is determined from the maximum and minimum processing rates.

Table 5.2: Input and output ranges for the continuous plant (Sources 1 - 3)

Source	**Total Input Mass** $[kg]$	**Total Output Mass** $[kg]$
1	150	66.0 - 76.5
2	170	74.8 - 86.7
3	60	26.4 - 30.6

Batch Plant

A unit by unit description of the batch plant is presented in this section. The processing rates and times for each task and the unit capacities are reported in Tables 5.3 and 5.4, respectively.

Unit 1: The first batch unit receives material from the 3 sources which are processed upstream in the continuous plant. In addition to this, material from the

fourth source enters the system at this unit. The amount of material processed in each lot is determined by the availability of the material and the maximum and minimum processing capacity for units in the batch plant and is thus left as a variable. The availability of material to the batch plant is restricted by the operation of the continuous plant. The maximum and minimum lot size differ for each source; this is a consequence of the different compositions of the sources and the different processing capacities of units in the batch plant.

A lot entering unit 1 must be composed of material from only one source. Once this lot enters the batch plant it may not be combined with any other material which is also being processed on the plant, not even material which originates from the same source. This 'no-mixing' rule is a result of a strict accounting procedure which takes place at the initiation and completion of each task. Each lot of material processed on this unit is divided into four fractions, namely F_1, F_2, F_3 and F_4, the values of which are reported in Table 5.5. These fractions are dependent on the composition of the material being processed, which is in turn dependent on the source from which the material originated. The four products of this task report to storage units $S1$, $S2$, $S3$ and $S4$.

Unit 2: The material from the storage unit $S1$ is processed in this unit. This material is further divided into two fractions, namely F_{1a} and F_{1b}, where $F_{1a} = 1 - F_{1b}$. The 'F_{1a}' portion of material reports to storage unit $S5$, where it is then passed on to either unit 5 or unit 6, depending on suitability. The 'F_{1b}' fraction of material reports to storage unit $S6$ and is processed further on unit 4 as task 4_1. The values of F_{1a} for each source are reported in Table 5.5.

Unit 3: Material from storage unit $S2$ is processed as task 3 on this unit. The material from this task is also divided into two fractions; F_{2a} and F_{2b}. The 'F_{2a}' portion of material is transfered to storage unit $S7$ and then to either unit 5 or 6. The other fraction is processed as task 4_2 on unit 4. Table 5.5 details the values of F_{2a} for each source.

Unit 4: This unit processes four tasks, where the order of processing these tasks is left as an optimization variable. The products from all these tasks are combined and stored in unit $S9$. This material is then passed onto unit 7, once all the products of the same lot are ready to be processed in this unit.

Unit 5 and 6: These two units operate in parallel and can process material from either unit 2 or 3. Material from unit 2 is processed as either task 5_1 on unit 5 or task 6_1 on unit 6, whereas the products from task 3 are processed as either tasks 5_2 or task 6_2 on units 5 and 6, respectively. For each lot that is processed through the plant only one task will happen on each of these units. For example, if task 5_1 takes place on unit 5, then task 6_2 will take place on unit 6 and the tasks 5_2 and 6_1 will not be required in the processing of that lot. The material from storage unit $S5$ is processed at a slower rate than material from $S7$ and thus it may be necessary to alternate the sequencing of material between Units 5 and 6, so as to prevent an accumulative bottleneck on any one unit. Due to the fact that different products may follow a different path through the plant, the plant may be considered to conform to a multipurpose flowshop.

Unit 7: Unit 7 is the last unit in the plant and involves the recombination of the lot that was originally processed on unit 1. The three tasks, 7_1, 7_2 and 7_3, take place on this unit and must occur simultaneously. Thus none of these tasks may begin until all the material from a particular lot has completed processing on the upstream units, namely units 4, 5 and 6.

Table 5.3: Processing times of tasks

Task	Dead Time $[min]$	Processing Rate $\left[\frac{min}{kg}\right]$
1	20	3.0
2	10	15.0
3	10	13.0
4_1	15	8.0
4_2	10	8.0
4_3	5	8.0
4_4	5	9.0
5_1/6_1	5	12.0
5_2/6_2	5	8.0
7_1/2/3	150	-

Table 5.4: Unit capacities

Unit	Minimum $[kg]$	Maximum $[kg]$
1	10.0	50.0
2	1.0	40.0
3	1.0	40.0
4	0.5	40.0
5	0.5	40.0
6	0.5	40.0
7	10.0	50.0

Table 5.5: Fractional composition of source material

	Source 1	Source 2	Source 3	Source 4
$\mathbf{F}_1$	0.2	0.5	0.3	0.1
$\mathbf{F}_2$	0.1	0.3	0.4	0.1
$\mathbf{F}_3$	0.3	0.1	0.1	0.6
$\mathbf{F}_4$	0.4	0.1	0.2	0.2
$\sum$	1.0	1.0	1.0	1.0
$\mathbf{F}_{1a}$	0.7	0.8	0.9	0.7
$\mathbf{F}_{2a}$	0.7	0.9	0.8	0.8

5.2 MILP Problem

The full problem is formulated as a mixed integer linear programming problem as was done in Chapters 3 and 4. The formulation is again based on the continuous-time formulation presented by Ierapetritou and Floudas (1998a).

The tasks 5_1 (5_2) and 6_1 (6_2) are essentially the same task, differing only with respect to the unit on which they are carried out. The reason why it is necessary to allow the material to switch between these units in the processing of different lots is to prevent an accumulative delay on either unit 5 or 6. To ensure these tasks are handled correctly on units 5 and 6, the following set of allocation constraints are employed, where an alternative notation is used for representing the indices of variables.

$$\begin{aligned}
\sum_{k \in K} (wv(5_1, k, n) + wv(5_2, k, n)) &\leq 1 \qquad \forall\, n \in N \\
\sum_{k \in K} (wv(6_1, k, n) + wv(6_2, k, n)) &\leq 1 \qquad \forall\, n \in N \\
\sum_{k \in K} (wv(5_1, k, n) + wv(6_1, k, n)) &\leq 1 \qquad \forall\, n \in N \\
\sum_{k \in K} (wv(5_2, k, n) + wv(6_2, k, n)) &\leq 1 \qquad \forall\, n \in N
\end{aligned} \tag{5.2}$$

The first two constraints ensure that the only a single task may be carried out on units 5 and 6 at any one time. While the second two constraints ensure that the same task is not carried out on both units. Thus either tasks 5_1 and 6_2 or tasks 5_2 and 6_1 may occur in the processing of any given batch. The inter-connectivity required to allow this operation is implemented quite simply in the material balance (see equation (2.20) on page 26).

To ensure the integrity of the model is maintained, the timing of these tasks is handled by implementing equation (2.26). These constraints are in addition to

the other timing constraints which are applicable to all tasks. An example of the implementation of this constraint for task 5_1, is presented in equation (5.3).

$$T^s(5\_1, 5, k, n+1) \geq T^f(5\_2, 5, k, n) - H(1 - wv(5, k, n)) \quad \forall\, k \in K, n \in N \tag{5.3}$$

This equation ensures that if the task 5_2 took place at an event point, then the starting time for task 5_1 at any later event point must be at least that of the completion time of task 5_2. This constraint is necessary because these tasks take place on the same unit. Note that this type of constraint is also applicable to the multiple tasks being carried out on Unit 4.

5.2.1 Solution of MILP Problem

Due to the addition of the six extra tasks the number of variables grew to 26 270, of which 1708 were integer variables. The number of integer variables is nearly triple that of the problem in Chapter 4. This is a consequence of the additional tasks and that each batch now requires the use of 6 event points as opposed to the 3 event points required for the problem in Chapter 4.

The optimization procedure was left to solve for a period of 72 hours, during which time no integer solution was found. The difficulty experienced in the solution procedure is attributed to the combination of following factors:

- The large number of both binary and continuous variables in the problem.
- The dependence of the processing times on the batch size.
- The frequent re-use of units.
- Presence of multiple units which can be used to process the same task.

The details of these issues are discussed in Chapter 3 and are thus not repeated here. However, in building up the problem of Chapter 4 to the full-scale industrial problem, the following observations were made.

- Initially the additional two tasks on unit 4 were added to the problem. At this stage Units 5 and 6 had not been incorporated. As expected, the inclusion of these tasks had a negative impact on the solution procedure.

  The solution was allowed to continue for a period of 12 hours, during which time a number of integral solutions were found. However, at the termination of the solution (due to an imposed time limitation) the integrality gap was in excess of 25%, indicating that the solution was far from completion.

  The difficulty experienced with obtaining a solution is attributed to the increase in the number of tasks processed on Unit 4. The additional two tasks increase the number of possible sequencing combinations from 2 (2!) to 24 (4!). Thus a total of 168 (24 combinations $\times$ 7 lots) sequencing combinations exist on Unit 4 for all 7 lots, as opposed to the 14 combinations for the problem in Chapter 4.

  Furthermore, the addition of two extra event points is required per lot in order to handle the inclusion of these tasks. Thus 140 (10 original tasks $\times$ 2 additional event points $\times$ 7 lots) integer variables were added to the problem, solely by increasing the number of event points. Now taking into account the two additional tasks on Unit 4, a further 66 (2 additional tasks $\times$ 33 event points) integer variables are added.

- The addition of Units 5 and 6 and hence tasks 5_1, 5_2, 6_1 and 6_2 resulted in a major deterioration in the solution procedure. Although no additional event points were required to incorporate these tasks into the problem, a further 132 (4 additional tasks $\times$ 33 event points) integer variables were added to the problem.

  However, the main reason for the decrease in efficiency of the solution procedure is attributed to the ability of both Units 5 and 6 to process the products from tasks 2 and 3. In this case the relaxed solution partially assigns multiple tasks to both of these units in order to reduce bottlenecking on any one

unit. The result is that the problem would exhibit a large integrality gap and solutions will only be found very deep into the tree-search.

As was mentioned previously, no solution to this problem was found after 72 hours of solution time. However, a note is made of the fact that the model used in this work yielded feasible results when the scheduling was determined *a priori* by fixing the values of the binary variables.

Due to the exceedingly long solution time required by this problem, this method of solution is not viable for practical purposes. There are alternative approaches that one may consider in order to obtain a solution in a realistic time frame. Alternative methods include using a discretized-time horizon as in the formulation proposed by Shah *et al.* (1993a) and the use of stochastic-type procedures such as genetic algorithms, simulated annealing and Monte-Carlo simulations. Although the solutions may well be generated in a smaller time, it is likely that the results obtained using these methods will result in sub-optimal solutions. Due to the nonlinear nature of the operation of the continuous plant, it is necessary to implement some sort of approximation technique, even in the continuous-time MILP formulation, in order to maintain linearity. Thus it is difficult to make a choice *a priori* as to which method will provide the most favourable results in terms of the quality of the solution and the time in which it is obtained.

Arguably the best solution procedure of the proposed alternative methods, in terms of the quality of the solution obtained, would be the discrete-time formulation. This is due to the fact that this method also relies on rigorous mathematical optimization to determine the solution. However, it is expected that the size of the problem would become very large in attempting to accurately approximate the batch-size dependent processing times of tasks. Furthermore, it is expected that much of the difficulties experienced in the solution procedure using the continuous-time formulation will be present in the discretized-time formulation. This is a consequence of the fact that many of the complexities which cause the solution procedure to deteriorate are due to the characteristics of the problem as opposed to the implemented formulation.

In addition to the above reasoning, the decision to use a random-type simulation was made, based on the fact that the GAMS model had already been formulated and could be adapted to handle the inclusion of randomly determined variables. Furthermore, it is possible to allow the solver to optimally determine the values of variable which are not determined by the random generator. A more detailed account of this procedure is presented in the following section.

5.3 Alternate Scheduling Approach

5.3.1 Random Simulation

In this section, two methods are presented in which the schedule is determined by using a series of randomly generated scenarios. These two methods differ in the extent upon which they rely on the random number generator to determine the value of certain variables. Carrying out these random simulations serves a dual purpose. Firstly they provide an alternative method of determining the schedule and also act as a benchmark against which the MILP problem can be compared. Unfortunately, no solution was obtained using the rigorous MILP approach and thus the random-type scenarios can only serve as an alternative scheduling procedure.

5.3.1.1 Method 1

One method of determining the schedule would be to determine the value of each variable in the simulation from a random number generator. Although it is possible to limit the value of variables between their upper and lower bounds, it was felt that determining the continuous variables from the random generator would result in a significantly large proportion of results being very poor. This is a consequence of there being infinitely many possible values that a continuous variable may take. Furthermore, as was shown in Scenario 2.9 (see page 84), by specifying the order in which the lots are processed and leaving the division of lots as a variable, the time required for solution is reduced considerably. This is due to the removal of

the majority of the binary variables from the problem. However, the operation of the continuous plant is not pre-determined and thus the problem remains a MILP problem. The same flexibility displayed in the continuous plant for the problem described in Chapter 4 is present in this method; this includes the allocation of material to event points, the time available for processing each task and the rate at which the material is processed.

Consequently, it was decided that by determining the sequence in which tasks occur *a priori* and allowing the solver to determine the optimal values for the continuous variables (such as the batch size), a large number of scenarios can be run in a relatively short time. Even though the solution time of such a scenario is likely to be far greater than that of a totally random generated scenario, the overall quality of the solutions should, in general, be far superior. It should be noted that this method of determining the schedule is better suited to problems where there exists a large degree of flexibility in the division of material into lots. Applying this method to problems where the processing times are independent of the batch size would not be as useful, since all flexibility of the problem would effectively be removed.

In this section, the order in which the lots are sequenced and the intra-lot scheduling on units 4, 5 and 6 is determined from a random number generator. The values of these variables were determined *a priori* using a FORTRAN script (see Section A.2 in the Appendices). A block diagram of the algorithm used to generate the random data is shown in Figure 5.2. The block diagram adheres to the standard format of block diagrams, where processes are shown as rectangles and decisions are shown as diamond-shaped figures. The encircled 'R' has been included to indicate the generation of a new random number. At the head of each arrow entering an encircled 'R', the seed is advanced and a new random number is generated. To ensure that there was no bias towards any particular result, a uniform random function was used to generate the random number.

A script file (see Section A.3) was written to execute the FORTRAN script and construct the GAMS input files for each set of randomly determined conditions. In a single execution of this script, 50 unique random conditions were determined and for each condition the optimization procedure was executed in GAMS. To ensure a

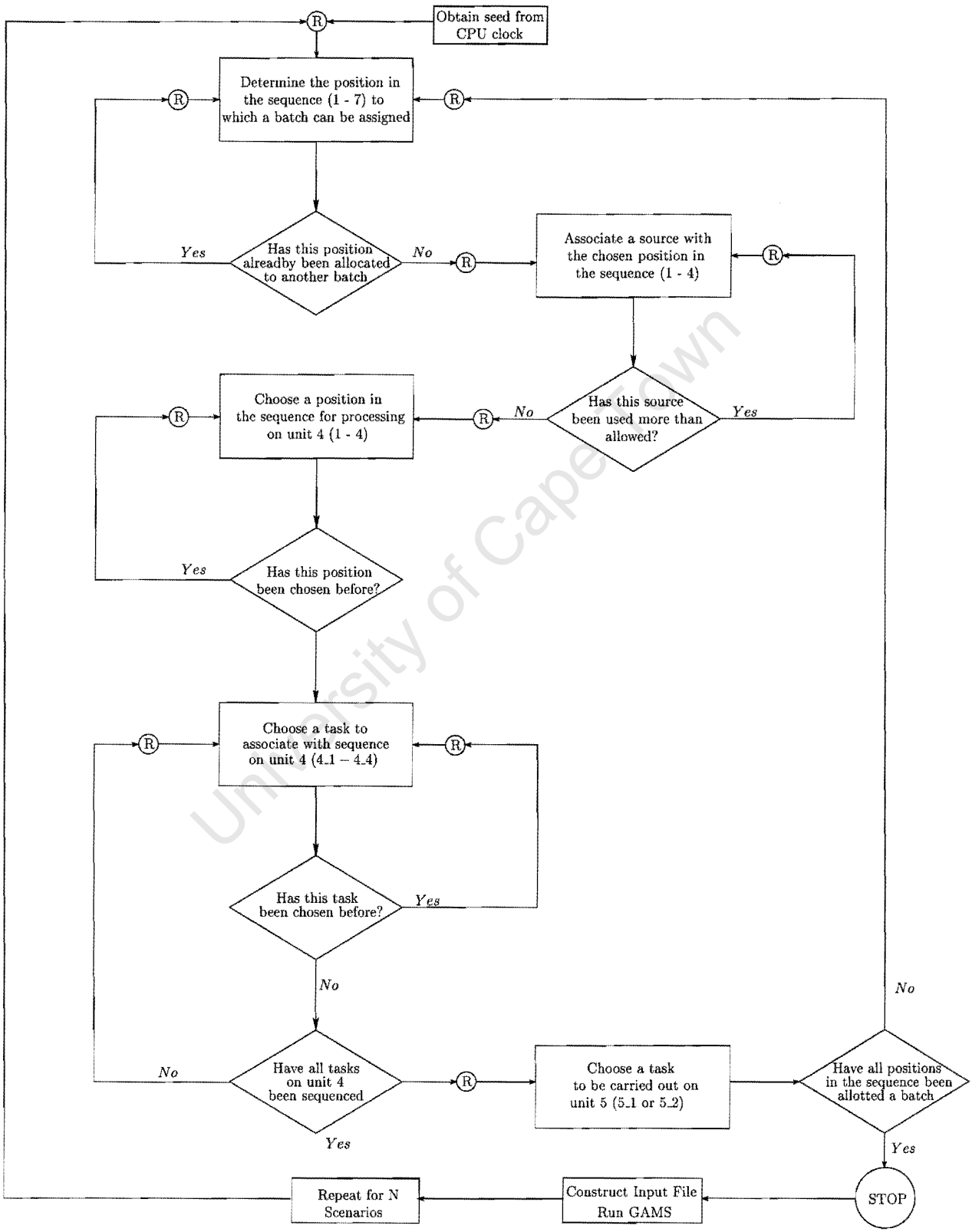


Figure 5.2: Block Diagram of Random Algorithm

different set of conditions for each time the script was executed, the first random number was seeded from the CPU clock.

This random number was then used to determine the order in which a batch will be sequenced *i.e.* between 1 and 7. If that number in the sequence had already been allocated to a batch then a new random number is generated. Once the order in which the lot will be sequenced has been determined, the next step is to determine from which source the material in that batch will be derived. Sources 1, 2 and 4 may be selected twice whereas source 3 may only make up a single batch. For each batch, the sequencing of the tasks on unit 4 is then determined, using similar logic. The task that takes place on unit 5 is determined next. Only a single task will take place on unit 5 for a single batch, the other task that could have possibly taken place on this task now takes place on unit 6. Once the intra-lot scheduling for a batch is complete the algorithm loops back to the start until all positions in the sequence have been allocated a batch. This procedure occurs 50 times to generate all the input files for the optimization procedure.

Results of Method 1

Running all 50 scenarios took approximately 7 hours. The resultant makespans for these simulations ranged between 2278 minutes and 2824 minutes, thus re-confirming that the order in which material is processed has a significant effect on the makespan.

The results presented here are for the best result of the 50 scenarios that were obtained using Method 1 to determine the schedule. The makespan for this scenario was 2278 minutes. Table 5.6 presents the ordering and size of the individual lots in the batch plant. An interesting result illustrated from this data is the fact that the first lot in the sequence is the maximum allowable batch size. In the scenario studies carried out in Chapter 3, the first batch processed was, in general, the minimum allowable size in order to prevent delays experienced at the beginning of the schedule. However, due to the fact that sources 1, 2 and 3 must be pre-processed in the continuous plant before entering the batch plant, a large batch of

material from source 4 can be processed initially since there is a delay before the other material becomes available.

Table 5.6: Results obtained for best scenario in Method 1

Order	1	2	3	4	5	6	7	Total
Source	4	2	1	2	1	3	4	
Mass	50.0	35.9	37.6	47.9	32.9	29.3	23.0	256.6

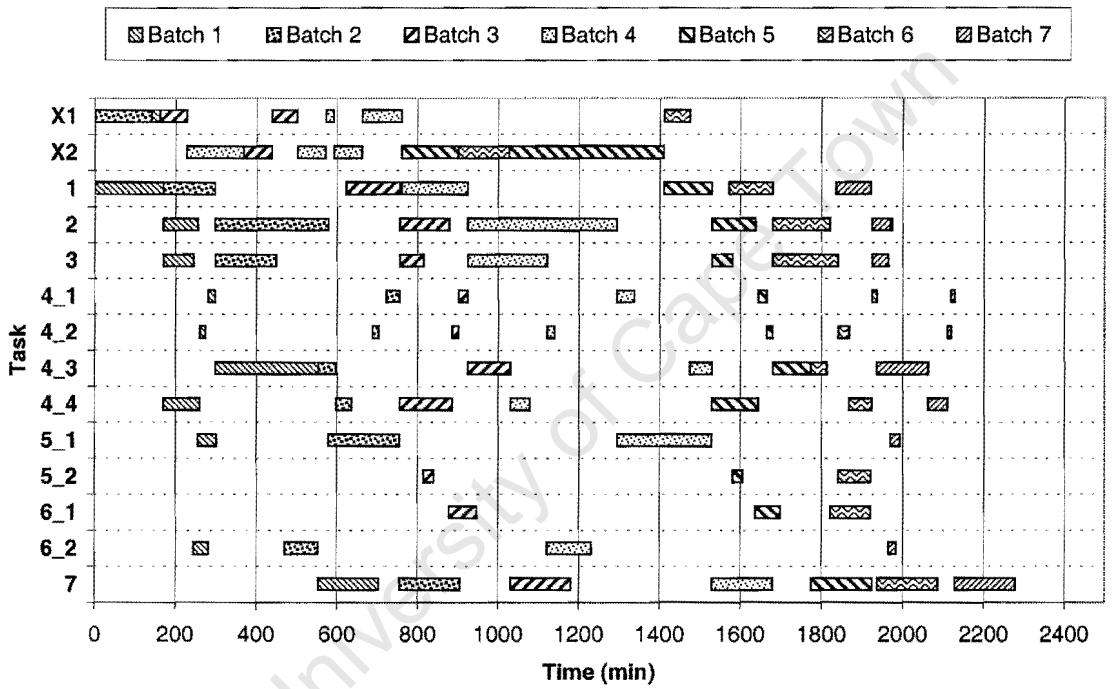


Figure 5.3: Results for Method 1

Figure 5.3 shows the manipulation of the continuous plant as tasks $X1$ and $X2$. Task $X1$ coincides with a slower operating rate but a more efficient operation than task $X2$, resulting in a smaller total amount of material being passed to the batch plant. Thus a trade off must be made between making the material available timeously (*i.e* without delaying processing in the batch plant) and minimizing the total amount of material that is passed to the batch plant. For example, in the last task that is processed (*i.e* Batch 7) there is ample time for pre-processing material on the continuous plant before it can be processed on the batch plant, thus this

material is processed as task $X1$ on the continuous plant to ensure the minimum amount of material is passed onto the batch plant.

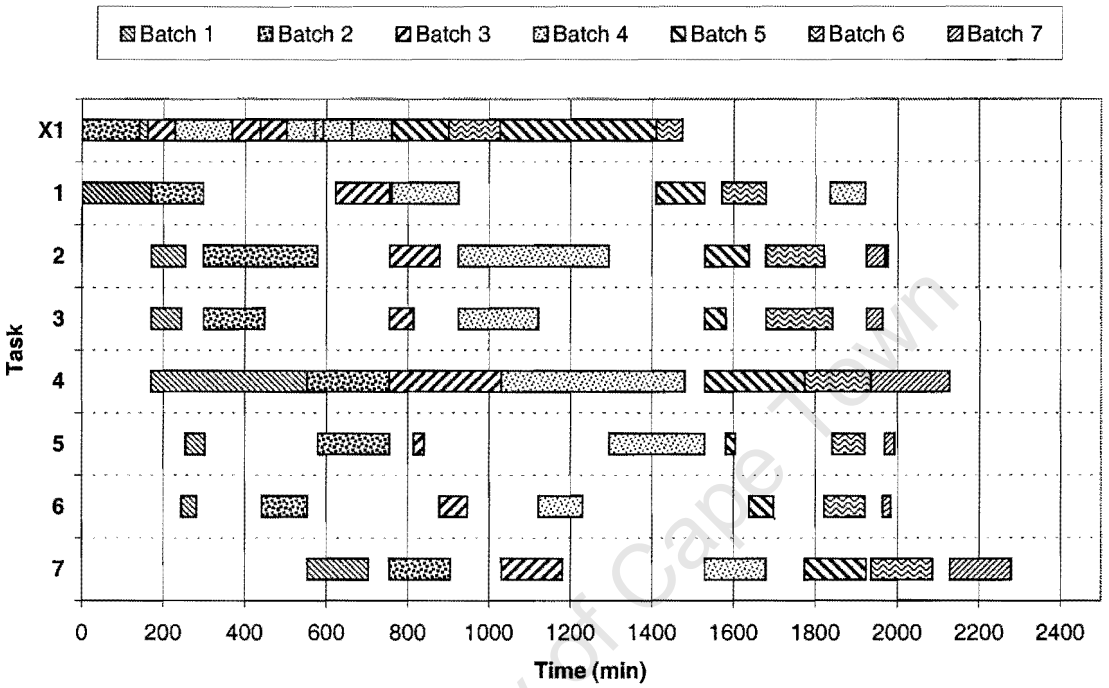


Figure 5.4: Results for Method 1 showing bottlenecking on units

Figure 5.4 is a Gantt chart showing the unit usage, as opposed to the processing of individual tasks (as in Figure 5.3). The presentation of the results in this manner allows one to easily locate the bottlenecks on the plant. The most obvious source of bottlenecking occurs at Unit 4. However, in many cases, the combined processing times of the tasks on Unit 2 and Unit 5 are equivalent to the processing time on Unit 4 (for batches 2 and 6). This illustrates how the optimization process subdivides material into lots so as to minimize the overall idle times on units, hence preventing large bottlenecks from occurring exclusively at one section of the plant.

5.3.1.2 Method 2

In order to determine the effect of discretizing the processing rate of the continuous plant, a series of random simulations were carried out in which the rate is determined

from a uniform random number. This randomly generated input data is used in addition to the data generated determining the lot sequencing, as in Section 5.3.1.1.

The primary aim of this scheduling procedure is to provide an alternative method for determining the operation of the continuous plant, while still maintaining the linearity of the problem. This method also provides a benchmark against which the level of accuracy of the discretized approximation for the operation of the continuous plant can be compared.

The processing rate must fall between the maximum and minimum processing rates of the continuous plant; this constraint is ensured by equation (5.4).

$$R_n = R_n^{min} + (R^{max} - R^{min}) \times U_n^{rand} \tag{5.4}$$

where:

R_n is the processing rate at event point n.

U_n^{rand} is a uniformly generated random number between 0 and 1 at event point n.

The portion of the material reporting to the product stream of the continuous unit is related to the rate and thus determined from equation (5.1) (see page 109). A new random number is generated for each event point in the schedule and hence a different processing rate is determined for each event point.

This solution-method differs to Method 1 in the manner in which the processing rate of the continuous plant is determined. In Method 1 the processing rate at each event point is determined by the solver from a set of discrete values. Whereas in Method 2, the value of the processing rate at each event point is determined *a priori* from a random generator. As was the case in Method 1 the problem is still a MILP problem, since the sequence of processing material on the continuous plant is left as a variable.

However, the flexibility of this sequencing is limited as a consequence of the pre-determined sequencing in the downstream batch plant. Thus the continuous plant

is required to provide at least the minimum amount of material necessary for a lot in the batch plant, without causing unnecessary delay. As mentioned previously, the minimum lot size is constrained by the minimum unit capacity.

However, there still remains a degree of flexibility associated with the operation of the continuous plant. Firstly, there are four event points between the processing of subsequent lots on the batch plant. During these event points the continuous plant may process between a maximum of four tasks and a minimum of zero tasks. Furthermore, the amount of time dedicated to the processing of each task in the continuous plant is left as a variable. Thus if the random generator allocates an 'unsuitable' processing rate to a particular event point, the continuous plant can either operate for a short period or not operate at all at that event point; this is determined by the optimization. Here, an 'unsuitable' processing rate refers to the situation where a different processing rate would be preferable considering the particular state of the plant at that time. Thus large processing times can be allocated to event points which have been assigned a more favourable processing rate, while the event points assigned less suitable processing rates can be allocated shorter processing times. Hence, the negative effects due to the rigidity introduced by the randomly determined operation of the continuous plant may be alleviated by the optimization.

The same procedure as in Section 5.3.1.1 was used to randomly determine lot sequencing and the intra-lot scheduling for this method. The difference is that within each execution of the GAMS model, 5 sub-problems were run. Thus a total of 5×50 scenarios are carried out in total. Figure 5.5 shows the structure of this algorithm in the form of a block diagram.

Within each sub-problem, a value for the processing rate of the continuous plant at each event point was determined using equation (5.4) and by calling the GAMS uniform random function. By running these sub-problems in a single GAMS execution a different set of random data is generated for each of these sub-problems. Thus each sub-problem involves a simulation with a different set of processing rates, but with the sequencing and intra-lot scheduling fixed for each set of 5 sub-problems.

To allow a fair comparison between the method of discretizing the processing rate

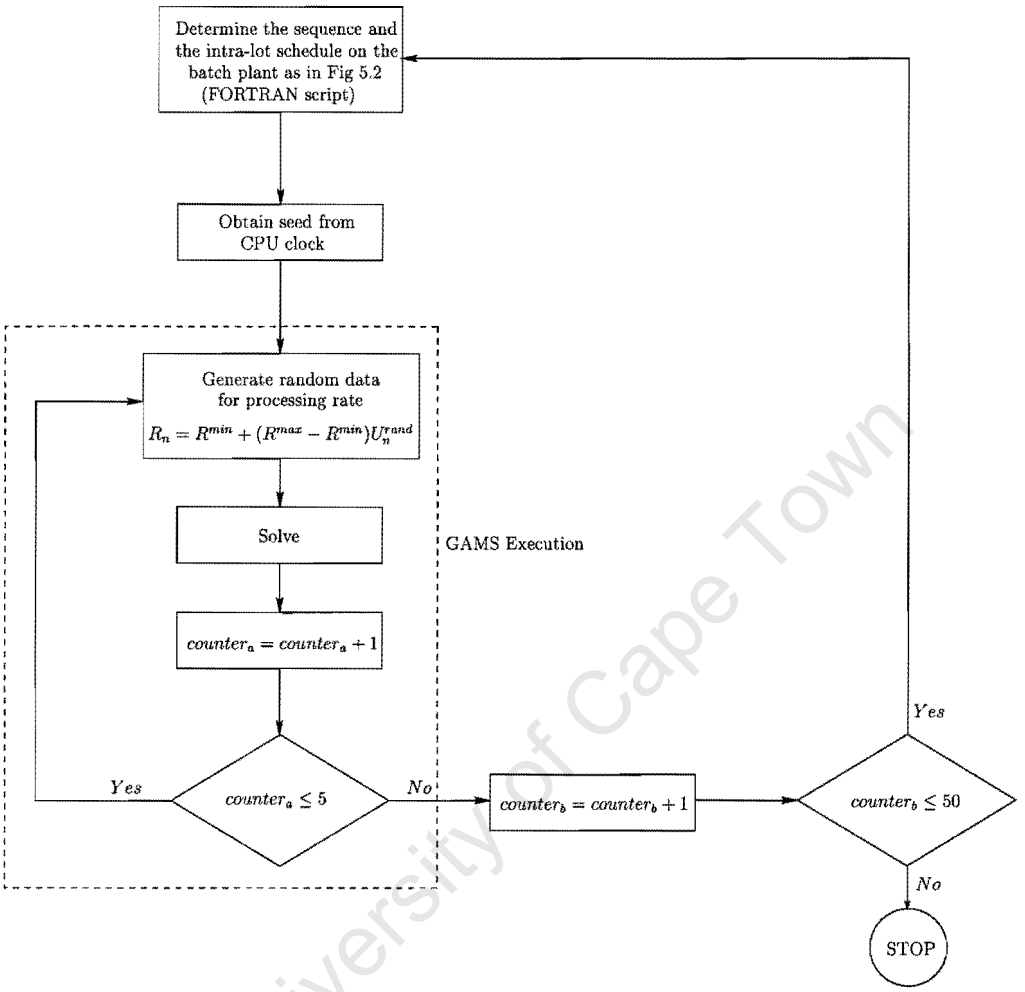


Figure 5.5: Block Diagram of Algorithm

(as in Section 5.3.1.1) and determining it randomly, it was necessary to ensure that the sets of data for each of the sub-problems were unique, not only within the set of sub-problems but also between all 50 scenarios. To maximize the likelihood that this was the case, the initial seed for each set of sub-problems was determined 'on-the-fly' by obtaining it from the CPU clock.

Results of Method 2

The time taken to complete all 250 scenarios was in the region of 18 hours. For the same conditions used in each of the 50 scenarios of Method 1, a further 5

sub-problems were completed for Method 2 in which the processing rate in each sub-problem was determined randomly. The results presented in this section correspond to those obtained for the same sequencing and intra-lot scheduling as in the particular scenario presented in the discussion of Method 1. Since there were 5 sub-problems performed for this particular scenario using Method 2, the results for the best of the 5 sub-problems is presented here. The makespan in this case was 2307 minutes, 29 minutes longer than the that obtained in Method 1. This result was in fact the best result of the 250 scenarios carried out using Method 2.

Table 5.7 details the order of lots in the processing sequence and their size. An interesting point is the fact that the total mass of material processed in this case is almost exactly that which was processed in the scenario obtained via Method 1. One would thus expect the makespans to differ by a smaller amount than 29 minutes, since the same sequencing and intra-lot scheduling is used in both scenarios. Due to the slow rate at which material from source 2 is initially processed on the continuous plant, the commencement of processing this task in the batch plant is delayed by 76 minutes.

However, on closer inspection it can be seen that the sizes of batches 4 and 5 differ significantly between these scenarios. This is a result of the manner in which the continuous plant is operated. Batch 4, which is made up of material from source 2 is and smaller in this case; whereas batch 5 which is derived from source 1, is larger relative to the results obtained using Method 1. This results in more favourable conditions, allowing the schedule to make up time and reduce the initial 76 minute delay to just 29 minutes.

Table 5.7: Results obtained for the best scenario Method 2

Order	1	2	3	4	5	6	7	Total
Source	4	2	1	2	1	3	4	
Mass	50.0	36.3	36.7	41.3	38.7	30.0	23.0	256.0

Figure 5.6a presents the results of this scenario in the form of a Gantt chart. In this figure the operation of the continuous plant is displayed as a single task, namely $X1$. This method of display was adopted due to the fact that the processing rate takes on a different value at each event point and hence the format adopted in

previous Gantt charts was not suitable. In order to present the processing rates at which the continuous plant operated, the tasks are labelled alphabetically (A-G). These tasks have been labelled alphabetically since they do not necessarily coincide with the ordering of batches 1-7 through the batch plant. The letters on this figure correspond to Figure 5.6b, where the different processing rates that are used for each task are shown as separate bars. For example, the first task on Figure 5.6a, labelled 'A' is processed at 3 different processing rates. The values of these rates are shown in Figure 5.6b by the first cluster of bars. The processing time for which the continuous plant is operated at each rate is shown at the head of the respective bar.

The Gantt chart (Figure 5.6a) is very similar to that obtained using Method 1 (Figure 5.3). There are however a number of subtle differences. These differences are mainly apparent in the operation of the continuous plant, where the order of processing and the rate at which it is processed differs. The timing of tasks in the batch plant are also very similar to those shown in Figure 5.3, except for the processing of batches 4 and 5 where the processing times differ.

Figure 5.6b presents the processing rates (as determined from the random generator) for the continuous plant. The value of these rates varies between 2.0 and 7.0 $\left[\frac{min}{kg}\right]$, as determined from equation (5.4). This figure illustrates the ability of the optimization procedure to manipulate the continuous plant in order to produce the most favourable overall result. For example, the 4^{th} task on the continuous plant, namely 'D', is carried out at two different rates. However, as can been seen from the respective times for which the plant operated at these rates, the majority of material is processed at a rate of 4.1 $\left[\frac{min}{kg}\right]$. Thus showing that even with the rigidity imposed by the rate being determined from a random number, there is still a degree of flexibility which allows the solver to allocate more processing time to more 'suitable' rates.

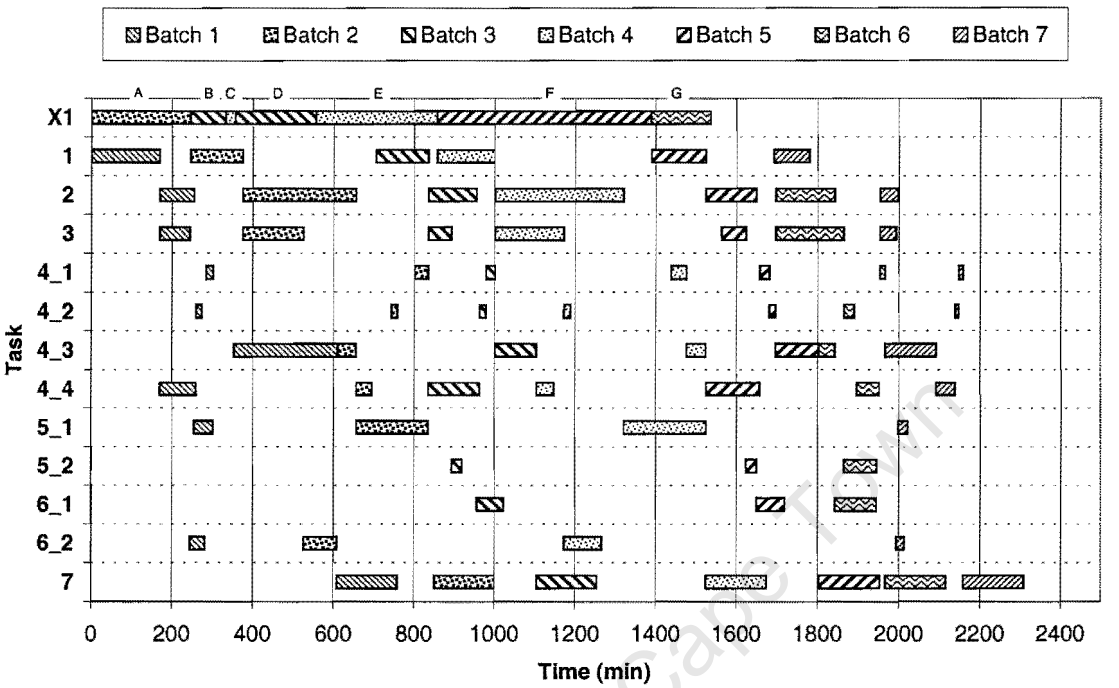


Figure 5.6a: Results for Method 2

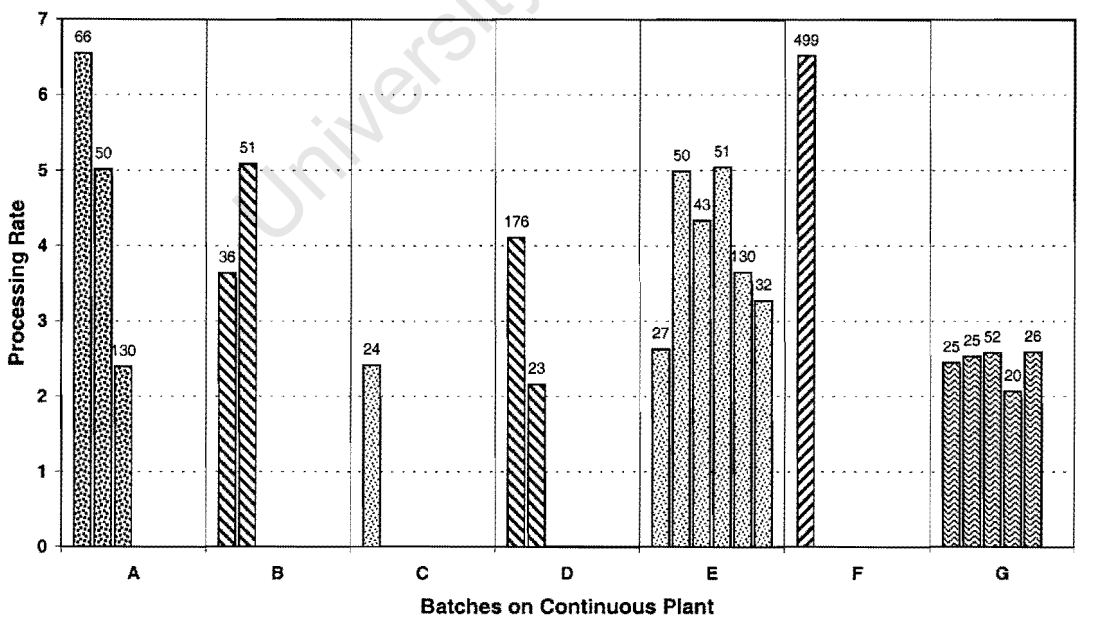


Figure 5.6b: Processing rates for the continuous plant

5.3.1.3 Comparison of Method 1 and Method 2

In this section a brief comparison on the quality of the solutions obtained using Methods 1 and Methods 2 is presented.

Figure 5.7 summarizes the results for the random scenario studies that were carried out using Methods 1 and 2, as described above. The solid boxes represent the value of the objective function for the results obtained using Method 1, for each of the 50 random scenarios. The transparent boxes represent the value of the best objective function obtained for the sub-problems of each of the 50 scenarios, using Method 2. The solid line represents the "best-value" obtained between these two methods and thus at least one of the data points will lie on this line for each scenario. Since the objective is to minimize the makespan, the smaller objective function values indicate a superior result. To avoid confusion between the data points from different scenarios, grid lines have been constructed to pass through the associated data points. Note, for clarity, the data has been sorted in ascending order with respect to the "best-value" line.

More often than not, the solid boxes fall on the "best-value" line. In fact only 11 of the 50 solid boxes are above the line. Moreover, in many cases the objective function obtained using Method 2 is significantly larger than was obtained using Method 1, however, the converse is not true. This indicates that the rigidity introduced with determining the processing rate of the continuous plant using a random generator, is too significant to overcome even with the flexibility still available to the continuous plant. Furthermore, by only carrying out 5 random sub-problems within each scenario, the usefulness of the random simulation is limited. However, running a larger number of sub-problems in each scenario would result in a significant increase in the overall solution time and will not necessarily provide a better result. This was found to be the case in a particular instance, where 25 additional sub-problems were run for the same scenario presented in the 'Results' section for Method 2. A more favourable solution was found than the best solution previously obtained using Method 2, however, is was still worse than the solution obtained using Method 1. Although the randomly generated processing rates (Method 2) can and sometimes do yield better results than Method 1, the results are obviously dependent

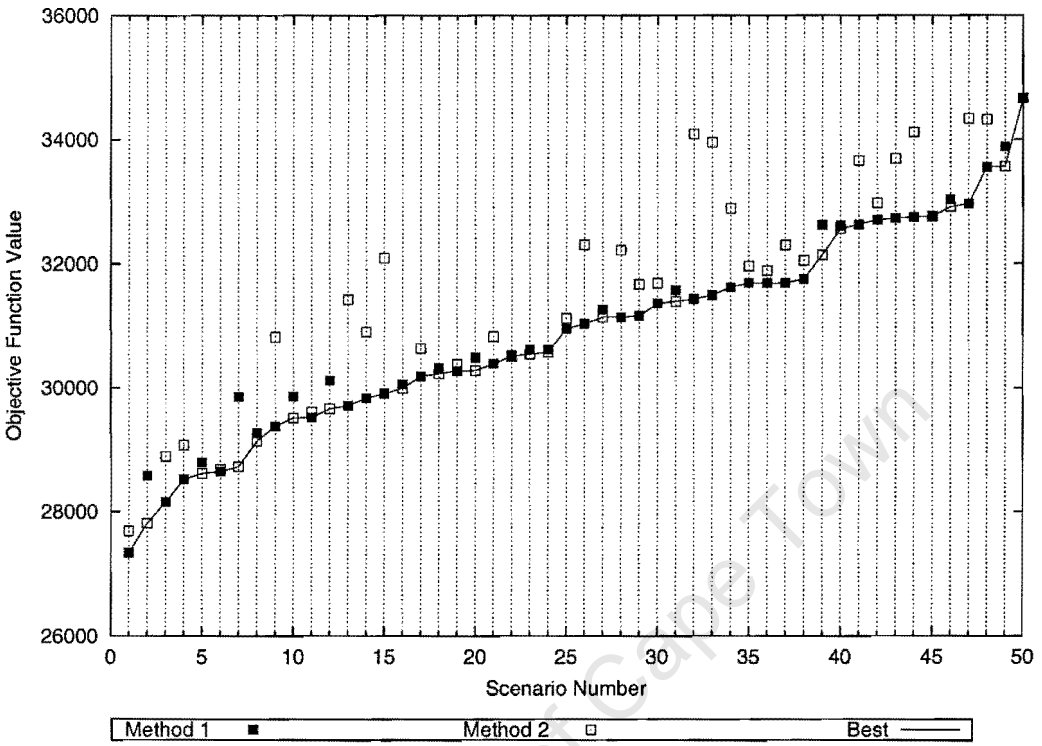


Figure 5.7: Comparison of the Results for Method 1 and Method 2

on the set of random values that the scenario is assigned. In general, it appears as if Method 1 is the better approach. This method can be further improved by increasing the level of discretization of the processing rate on the continuous plant, although this will impact negatively on the solution time.

The ultimate evaluation of the quality of the approximation of the method where the processing rate is discretized, is to compare the results against those obtained where the processing rate is left as a continuous function. However, as discussed previously, this introduces a nonlinearity into the problem thus resulting in a mixed-integer nonlinear programming problem. The GAMS model was easily adapted to incorporate the model as a continuous function, however, no solution to the problem was obtained after over 15 hours of solution time, even with the sequencing and intra-lot scheduling determined *a priori*. The reason for this is unknown but it is suspected that it is due to the problem being poorly scaled.

Concluding Remarks

This chapter addressed the full-scale industrial problem. The resulting problem proved to be too complex to solve within a 'reasonable' time using the continuous-time formulation that was implemented. Subsequently, alternative methods were posed and work was carried out using two methods which were based on running multiple scenarios which were partially determined using a random number generator.

These two methods proved to be reasonably successful, although the solutions times were possibly a bit too slow to be of much practical use. However, through inspection one could eliminate many combinations that consistently result in poor solutions by manipulating the possible combinations that can be randomly generated in the FORTRAN script, thus increasing the overall quality of the solutions and allowing 'superior' orders to be analysed. This in effect reduces the number of scenarios that must be run in order to obtain good solutions.

Chapter 6

Conclusions

The use of optimal scheduling is motivated by manufacturer's need to reduce processing costs and to meet customer demands. Batch processes are particularly amenable to the manipulation of the scheduling of tasks as a result of the inherent flexibility exhibited by these processes.

In this thesis, the scheduling of an integrated batch and continuous processing plant was investigated. This involved the formulation of the plant model as a mixed-integer linear programming problem. The formulation implemented in this thesis was based on the continuous-time formulation proposed by Ierapetritou and Floudas (1998a). This formulation was preferred over the discrete-time formulation because it does not require the approximation of the processing times of tasks using discrete intervals. Furthermore, the ease and accuracy with which the continuous-time formulation lends itself to in incorporating batch-size dependent processing times into the model, makes it the preferred formulation for this particular problem.

In order to model the proposed problem, a number of additional constraints were included into the formulation. These constraints incorporated the handling of the 'no-mixing' policy, the simultaneous recombination of material and batch-size dependent processing times.

In Chapters 3 and 4 a number of scenarios was carried out. These scenarios served a dual purpose in that they elucidate the ability of the formulation to optimize the operation of the plant as well as illustrate the flexibility of the formulation in handling various situations (such as unit unavailability).

A primary aim of this thesis was to attempt to determine the scheduling of an industrial problem using rigorous mathematical optimization. Although this aim was not strictly achieved, an alternative scheduling procedure was presented. The alternative method maintained some of flexibility of the original problem, thereby allowing the solver room for manipulation. However, solutions obtained using this method cannot be guaranteed to be optimal.

The inability of the MILP approach to locate an integral solution for the full-scale industrial plant was attributed to a number of factors. Firstly, the large number of binary (1708) and continuous (26270) variables in the problem give an indication of its magnitude and complexity. A particular characteristic of the problem was the fact that the processing times of the majority of tasks were dependent on the batch size. This resulted in the problem exhibiting a large integrality gap and integer solutions only being located very deep in the branch and bound search. The suitability of units to process multiple tasks was found to impact negatively on the computational efficiency of the problem. This is the case for Unit 4, where four tasks must be processed on this unit during each lot. This characteristic results in the relaxed problems in the branch and bound search, partially assigning multiple tasks to occur simultaneously. The result of this is that the problem exhibits a large integrality gap due the fractional values of integer variables in the relaxed solution. A further consequence of this is that integer solution are located deep in the tree-search, thus requiring the investigation of a large number of nodes to locate integral solutions and to verify the optimal solution. The same issues are evident for the parallel processing of tasks on unit 5 and 6.

The combination of these factors rendered the problem too complex to solve within a practical time frame. Hence alternate scheduling methods were proposed. The alternate methods that were presented in this thesis centred around reducing the size and complexity of the problem by setting the value of certain variables using

a random number generator. This in effect reduced the number of binary variables in the problem significantly and enabled solutions to be obtained in a short time. Thus it was possible to run a series of random scenarios in order to locate a meaningful result. There were however, a number of integer variables remaining in the formulation of the proposed random methods and thus the problem was solved as a MILP problem.

The adaption of the full MILP model, which had already been formulated, to implement the proposed method was a straight forward task. However, the investigation of the other proposed methods such as the use of genetic algorithms or simulated annealing to refine the stochastic search may well provide more favourable results.

A point which was highlighted in the scenario study, carried out in Chapter 3, was the impact which inaccurate plant data can exert on the solution obtained. The plant data which is used to determine the scheduling is generally based on measured averages. Thus it is unlikely that the data is an accurate reflection of the true situation on the plant at any one time. The result is that the optimal schedule as determined using rigorous optimization techniques is in fact sub-optimal for the actual plant. This emphasizes the need for accurate data to be made available so as to allow the mathematical model to accurately represent the plant. This however may not be practical and furthermore the operation of the plant may still deviate from the schedule due to unpredictable occurrences. This necessitates the use of an online method which can both alleviate or take advantage of deviations.

The following recommendations are made with respect to other issues in the extension of the work carried out in this thesis. Firstly the investigation of alternate methods; this includes the implementation of the model using the discrete-time formulation as proposed by Shah *et al.* (1993a), using genetic algorithms and the simulated annealing approach to solve the problem. Another area where there is scope for improvement is the manipulation of the branch and bound method to exploit the characteristics of scheduling problems. Shah *et al.* (1993b) address this issue and propose methods which improve the efficiency of the algorithm for scheduling problems.

Another area in which there is scope for improvement is in the extension of scheduling to incorporate the online problem. Various reactive scheduling mechanisms have been presented in the literature (Cott and Macchietto, 1989), which address the issues of implementing and controlling the plant scheduling. However, there is still scope in the application of both traditional and advanced control methods to control the plant scheduling.

References

Brooke, A., Kendrick, D., and Meeraus, A. (1988). *GAMS: A Users Guide.* Scientific Press.

Chen, X., Rao, S., Yu, J., and Pike, R. (1996). Comparison of GAMS, AMPL and MINOS for Optimization. *Chem. Eng. Education*, 30(3):220–227.

Cott, B. and Macchietto, S. (1989). Minimizing the Effects of Batch Process Variabiltiy using On-Line Schedule Modification. *Computers Chem. Engng*, 13(1/2):105–113.

CPLEX Optimization Inc. (1995). *Using the CPLEX Callable Library.* Incline Village, NV, USA.

Egli, U. and Rippin, D. (1986). Short-Term Scheduling for Multiproduct Batch Chemical Plants. *Computers Chem. Engng*, 10:303–325.

Floudas, C. (1995). *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications.* Oxford University Press.

Fourer, R., Gay, D., and Kernigham, B. (1993). *AMPL: A Modelling Language for Mathemtical Programming.* Scientific Press.

GAMS Development Corporation (1999). *GAMS - The Solver Manuals.* Washington, DC, USA.

Glover, F. (1975). Improved Linear Integer Programming Formulations of Nonlinear Integer Problems. *Manage. Sci.*, 22:455–460.

Ierapetritou, M. and Floudas, C. (1998a). Effective Continuous-Time Formulation for Short-Term Scheduling. 1. Multipurpose Batch Processes. *Ind. Eng. Chem. Res.*, 37:4341–4359.

Ierapetritou, M. and Floudas, C. (1998b). Effective Continuous-Time Formulation for Short-Term Scheduling. 2. Continuous and Semicontinuous Processes. *Ind. Eng. Chem. Res.*, 37:4360–4374.

Ierapetritou, M. and Floudas, C. (1999). Effective Continuous-Time Formulation for Short-Term Scheduling. 3. Multiple Intermediate Due Dates. *Ind. Eng. Chem. Res.*, 38:3446–3461.

Ishii, N. and Muraki, M. (1996). A Process-Variability-Based Online Scheduling System in Multiproduct Batch Proces. *Computers Chem. Engng*, 20:217–234.

Karimi, I. A. and Reklaitis, G. (1985). Deterministic Variability for Intermediate Storage in Non-Continuous Processes. *AICHE Jl*, 31.

Kondili, E., Pantelides, C., and Sargent, R. (1993). A General Algorithm for Short-Term Scheduling of Batch Operations - I. MILP Formulation. *Computers Chem. Engng*, 17:211–227.

Ku, H., Rajagopalan, D., and Karimi, I. (1987). Scheduling in Batch Processes. *Chem. Engng Prog.*, 83:35–45.

Mah, R. (1990). *Chemical Process Structures and Information Flows.* Butterworth.

Mauderli, A. and Rippin, D. (1979). Production Planning and Scheduling for Multipurpose Batch Chemical Plants. *Computers Chem. Engng*, 13:199–206.

Mockus, L. and Reklaitis, G. (1997). Mathematical Programming Formulation for Scheduling of Batch Operations based on Nonuniform Time Discretization. *Computers Chem. Engng*, 21(10):1147–1156.

Moon, S. and Hrymak, A. (1999). Scheduling of the Batch Annealing Process - Deterministic Case. *Computers Chem. Engng*, 23:1193–1208.

Pantelides, C. (1994). Unified Frameworks for Optimal Process Planning and Scheduling. *Foundations of Computer-Aided Process Operations*, CACHE:253–275.

Papageorgiou, L. and Pantelides, C. (1996). Optimal Campaign Planning/Scheduling of Multipurpose Batch Semicontinuous Plants. 1. Mathematical Formulation. *Ind. Eng. Chem. Res.*, 35:488–509.

Rich, S. and Prokopakis, G. (1986). Scheduling and Sequencing of Batch Operations in a Multipurpose Plant. *Ind. Eng. Chem. Process. Des. Dev.*, 25:979–987.

Roslof, J., Harjunkoski, I., T.Westerlund, and J.Isaksson (1999). A Short-Term Scheduling Problem in the Paper-Converting Industry. *Computers Chem. Engng Suppl.*, pages S871–S874.

Schilling, G. and Pantelides, C. (1996). A Simple Continuous-Time Process Scheduling Formulation and a Novel Solution Algorithm. *Computers Chem. Engng Suppl.*, 20:S1221–S1226.

Schilling, G. and Pantelides, C. (1999). Optimal Periodic Scheduling of Multipurpose Plants. *Computers Chem. Engng*, 23:635–655.

Shah, N. and Pantelides, C. (1991). Optimal Long-Term Campaign Planning and Design of Batch Opertions. *Ind. Eng. Chem. Res.*, 30:2308–2321.

Shah, N., Pantelides, C., and Sargent, R. (1993a). A General Algorithm for Short-Term Scheduling of Batch Operations - II. Computational Issues. *Ind. Eng. Chem. Res.*, 17:229–244.

Shah, N., Pantelides, C., and Sargent, R. (1993b). Optimal Periodic Scheduling of Multipurpose Batch Plants. *Annals of Oper. Res.*, 42:193–228.

Suhami, I. and Mah, R. (1981). An Implicit Enumeration Scheme for the Flowshop Problem with No Intermediate Storage. *Computers Chem. Engng*, 5:83–91.

Wellons, M. and Reklaitis, G. (1991). Scheduling of Multiproduct Batch Chemical Plants: 1. Formulation of Single-Product Campaigns. *Ind. Eng. Chem. Res.*, 30:671–688.

Willams, H. (1993). *Model Building in Mathematical Programming.* Wiley, 3rd edition.

Yee, K. and Shah, N. (1998). Improving the Efficiency of Discrete Time Scheduling Formulation. *Computers Chem. Engng Suppl.*, 22:S403–S410.

Zhang, X. and Sargent, R. (1994). The Optimal Operation of Mixed Production Facilities - A General Formulation and some Approaches for the Solution. In *5th Int. Symp. on Proc. Sys Engng*, pages 171–177, Kyongju, Korea.

Appendix A

Source Code

A.1 GAMS Code

This code was used to determine the scheduling using the GAMS random function 'uniform' to generate random data to determine the processing rate on the continuous plant. The majority of equations are written out in full form so as to make the code clearer. Some of the equations, such as the 'different tasks on different units' and 'different task on same units' constraints, result in a degree of repetition in order to cover all eventualities. However, this is not a problem since the CPLEX presolve eliminates redundencies from the problem. The objective function used in this case is the minimization of the makespan.

GAMS - SAMPLE SCHEDULING CODE

```
$OFFSYMXREF OFFSYMLIST OFFUELLIST OFFUELXREF OFFDIGIT OFFLISTING
*UNIT C2 ignored for this scenario
OPTION LIMROW = 0
OPTION LIMCOL = 0
OPTION SOLPRINT = Off
OPTION SYSOUT = OFF

SETS
    I       Tasks from Source a     /C1,C2,1*3,4_3,4_4,4_1,4_2,5_1,5_2,6_1,6_2,7_1*7_3/
    J       Units                   /C1,C2,1*7/
    K       Batches                 /1*4/
    S       Storage                 /S1_1*S1_4,S2*S13,S20/
    N       Event Point             /N0*N32/
    M       No of rates             /1*2/
    F       Fractions               /F1*F4/
    F2      Unit 2 3                /Fi,Fii/;

ALIAS(N,nn);
```

```
ALIAS(N,ni);
ALIAS(I,ii);
ALIAS(S,ss);
ALIAS(K,kk);

SET Ite(N,nn);
Ite(N,nn)$(ORD(nn) LE ORD(N)) = yes;

* tasks 4_3 and 4_4 ie not via unit 2 and 3
SET UNIT4a(I);
UNIT4a(I)$((ORD(I) >= 6)AND(ORD(I) <= 7)) = YES;
* tasks 4_1 and 4_2
SET UNIT4b(I);
UNIT4b(I)$((ORD(I) >= 8)AND(ORD(I) <= 9)) = YES;
SET UNIT4(I);
UNIT4(I)$((ORD(I) >= 6)AND(ORD(I) <= 9)) = YES;
SET UNIT5(I);
UNIT5(I)$((ORD(I) >= 10)AND(ORD(I) <= 11)) = YES;
SET UNIT6(I);
UNIT6(I)$((ORD(I) >= 12)AND(ORD(I) <= 13)) = YES;
SET UNIT56(I);
UNIT56(I)$((ORD(I) >= 10)AND(ORD(I) <= 13)) = YES;
SET UNIT7(I);
UNIT7(I)$((ORD(I) >= 14)AND(ORD(I) <= 16)) = YES;

PARAMETERS
a(M)  Rate Values /1  2
                  2  7/;
SCALARS
     bi, c,
     H              Time horizon                          hr      /3000/;

bi = (0.51 -0.44) / (a('1') - a('2'));
c = 0.44 - bi*a('2');

PARAMETERS
FR(N),p(N),
STIN(K)   Mass of each batch available for processing into continuous plant
             /1 150
              2 170
              3 60
              4 0/

STIN1(K)   Mass of each batch available for processing into batch plant
             /1 0
              2 0
              3 0
              4 73/;

TABLE  SPLIT(F,K)
                    1     2     3     4
               F1   0.2   0.5   0.3   0.1
               F2   0.1   0.3   0.4   0.1
               F3   0.3   0.1   0.1   0.6
               F4   0.4   0.1   0.2   0.2;

TABLE  SPLIT2(F2,K)
                     1     2     3     4
               Fi    0.3   0.2   0.1   0.3
               Fii   0.3   0.1   0.2   0.2;

VARIABLES
    TTime
    Z
    R(N);

POSITIVE VARIABLES
    B(I,J,K,N)          Amount of material undertaking task "i" in unit "j at event point "n"
    ST(S,N)             Storage unit S at event point N
    Si(K,N)             Storage units for Source k at event point N
    D(S,K,N)            Amount removed from the system
    Tf(I,J,K,N)         Finish Time of Task "i" on Unit "j" at event point "n"
    Ts(I,J,K,N)         Start Time of Task "i" on Unit "j" at event point "n"
    Total               Total Amount of material removed
    TotalUS             Total amount of Undersize material
    TotalOS             Total amount of Oversize material
```

```
    Total              Total Amount of material removed from system
    Total1             Total Amount of material processed in unit 1
    Total2             Total Amount of material processed in unit 2
    Total3             Total Amount of material processed in unit 3
    Total4             Total Amount of material processed in unit 4
    Total5             Total Amount of material processed in unit 5;

BINARY VARIABLES
    wv(I,K,N)    Task i at event point Ni
    yv(J,N)    Unit j at event point Ni;

EQUATIONS

A001(N),A101(N),A102(N),A103(N),A104(N),A105(N),A106(N),A107(N),A108(N),A109(K,N),A110(N),A111(N),
A201(K,N),A202(K,N),A203(K,N),A204(K,N),A205(K,N),A206(K,N),A207(K,N),A208(K,N),A209(K,N),
A210(K,N),A211(K,N),A212(K,N),A301(K,N),A302(K,N),A303(K,N),A304(K,N),A305(K,N),A306(K,N),
A307(K,N),A308(K,N),A309(K,N),A310(K,N),A311(K,N),A312(K),A313(K),A401(I,K,N),A500(N),A501(N),
A502(N),A503(N),A504(N),A505(N),A506(N),A507(N),A508(N),CCi(K,N),CCii(K,N),CC01(K,N),CC02(K,N),
CC03(K,N),CC04(K,N),CC05(K,N)CC06(K,N),CC07(K,N),CC08(K,N),CC09(K,N),CC10(K,N),CC11(K,N),
CC12(K,N),CC13(K,N),CC14(K,N),CC001(K,N),CC002(K,N),CC101(K,N),CC102(K,N),CC103(K,N),CC104(K,N),
CC105(K,N),CC106(K,N),CC107(K,N),CC108(K,N),CC108(K,N),CC109(K,N),CC110(K,N),CC111(K,N),
CC112(K,N),CC113(K,N),CC114(K,N),SC1(S,N),SC2(K,N),SC3(K,N),SC4(K,N),SC5(K,N),SC6(K,N),SC7(N),
SC8(N),SC9(N),SC10(N),SC11(N),SC12(N),SC13(N),MB0(K,N),MB1(S,K,N),MB2(N),MB3(N),MB4(N),MB5(N),
MB6(N),MB7(N),MB8(N),MB9(N),MB10(N),MB11(N),MB12(N),MB20(N),DC(I,J,K,N),DCi(K,N),DCii(K,N),
DC01(K,N),DC02(K,N),DC03(K,N),DC04(K,N),DC05(K,N),DC06(K,N),DC07(K,N),DC08(K,N),DC09(K,N),
DC10(K,N),DC11(K,N),DC12(K,N),DC13(K,N),DC14(K,N),STSU01(K,N),STSU02(K,N),STSU03(K,N),STSU04(K,N),
STSU05(K,N),STSU06(K,N),STSU11(K,N),STSU12(K,N),STSU13(K,N),STSU21(K,N),STSU22(K,N),STSU23(K,N),
STSU31(K,N),STSU32(K,N),STSU33(K,N),STSU41(K,N),STSU42(K,N),STSU43(K,N),STSU44(K,N),STSU45(K,N),
STSU46(K,N),STSU47(K,N),STSU48(K,N),STSU49(K,N),STSU4a(K,N),STSU4b(K,N),STSU4c(K,N),STSU51(K,N),
STSU52(K,N),STSU53(K,N),STSU54(K,N),STSU55(K,N),STSU56(K,N),STSU61(K,N),STSU62(K,N),STSU63(K,N),
STSU64(K,N),STSU65(K,N),STSU66(K,N),STSU71(K,N),STSU72(K,N),STSU73(K,N),STSU74(K,N),STSU75(K,N),
STSU76(K,N),STSU77(K,N),STSU78(K,N),STSU79(K,N),DTSU01(K,N),DTSU02(K,N),DTSU03(K,N),DTSU04(K,N),
DTSU05(K,N),DTSU06(K,N),DTSU07(K,N),DTSU08(K,N),DTSU11(K,N),DTSU12(K,N),DTSU13(K,N),DTSU14(K,N),
DTSU21(K,N),DTSU22(K,N),DTSU23(K,N),DTSU24(K,N),DTSU31(K,N),DTSU32(K,N),DTSU33(K,N),DTSU34(K,N),
DTSU41(K,N),DTSU42(K,N),DTSU43(K,N),DTSU44(K,N),DTSU45(K,N),DTSU46(K,N),DTSU47(K,N),DTSU48(K,N),
DTSU401(K,N),DTSU402(K,N),DTSU403(K,N),DTSU404(K,N),DTSU405(K,N),DTSU406(K,N),DTSU407(K,N),
DTSU408(K,N),DTSU51(K,N),DTSU52(K,N),DTSU53(K,N),DTSU54(K,N),DTSU55(K,N),DTSU56(K,N),DTSU57(K,N),
DTSU58(K,N),DTSU61(K,N),DTSU62(K,N),DTSU63(K,N),DTSU64(K,N),DTSU65(K,N),DTSU66(K,N),DTSU67(K,N),
DTSU68(K,N),DTSU71(K,N),DTSU72(K,N),DTSU73(K,N),DTSU74(K,N),DTSU75(K,N),DTSU76(K,N),DTSU77(K,N),
DTSU78(K,N),DTSU79(K,N),DTSU710(K,N),DTSU711(K,N),DTSU712(K,N),DTSU001(K,N),DTSU002(K,N),
DTSU003(K,N),DTSU004(K,N),DTSU005(K,N),DTSU006(K,N),DTSU007(K,N),DTSU008(K,N),DTSU411(I,K,N),
DTSU412(I,K,N),DTSU413(I,K,N),DTSU414(I,K,N),DTSU415(I,K,N),DTSU416(I,K,N),DTSU417(I,K,N),
DTSU418(I,K,N),DTSU419(I,K,N),DTSU420(I,K,N),DTSU422(I,K,N),DTSU423(I,K,N),DTSU424(I,K,N),
DTSU425(I,K,N),DTSU426(I,K,N),DTSU427(I,K,N),DTSU511(K,N),DTSU512(K,N),DTSU513(K,N),DTSU514(K,N),
DTSU515(K,N),DTSU516(K,N),DTSU517(K,N),DTSU518(K,N),DTSU611(K,N),DTSU612(K,N),DTSU613(K,N),
DTSU614(K,N),DTSU615(K,N),DTSU616(K,N),DTSU617(K,N),DTSU618(K,N),DTDU11(K,N),DTDU12(K,N),
DTDU21(K,N),DTDU31(K,N),DTDU411(K,N),DTDU421(K,N),DTDU431(K,N),DTDU441(K,N),DTDU511(K,N),
DTDU521(K,N),DTDU611(K,N),DTDU621(K,N),DTDU7a1(K,N),DTDU7a2(K,N),DTDU7b1(K,N),DTDU7b2(K,N),
DTDU7c1(K,N),DTDU7c2(K,N),DTDU7c3(K,N),DTDU7c4(K,N),DTDU201(K,N),DTDU202(K,N),DTDU203(K,N),
DTDU204(K,N),DTDU301(K,N),DTDU302(K,N),DTDU303(K,N),DTDU304(K,N),DTDU401(I,K,N),DTDU402(I,K,N),
DTDU403(I,K,N),DTDU404(I,K,N),DTDU405(I,K,N),DTDU406(I,K,N),DTDU407(I,K,N),DTDU408(I,K,N),
DTDU4a1(K,N),DTDU4a2(K,N),DTDU4a3(K,N),DTDU4a4(K,N),DTDU4a5(K,N),DTDU4a6(K,N),DTDU4a7(K,N),
DTDU4a8(K,N),DTDU501(K,N),DTDU502(K,N),DTDU503(K,N),DTDU504(K,N),DTDU505(K,N),DTDU506(K,N),
DTDU507(K,N),DTDU508(K,N),DTDU601(K,N),DTDU602(K,N),DTDU603(K,N),DTDU604(K,N),DTDU605(K,N),
DTDU606(K,N),DTDU607(K,N),DTDU608(K,N),DTDU701(K,N),DTDU702(K,N),DTDU703(K,N),DTDU704(K,N),
DTDU705(K,N),DTDU706(K,N),DTDU707(K,N),DTDU708(K,N),DTDU709(K,N),DTDU710(K,N),DTDU711(K,N),
DTDU712(K,N),DTDU713(K,N),DTDU714(K,N),DTDU715(K,N),DTDU716(K,N),DTDU717(I,K,N),DTDU718(I,K,N),
DTDU719(I,K,N),DTDU720(I,K,N),CPT1(I,J,K,N),CPT2(I,K,N),CPT3(I,K,N),CPT4(I,K,N),CPT5(K,N),
CPT6(K,N),CPT7(K,N),CPT8(K,N),CPT9(K,N),TH01(K,N),TH02(K,N),TH03(K,N),TH04(K,N),TH11(K,N),
TH12(K,N),TH21(K,N),TH22(K,N),TH31(K,N),TH32(K,N),TH41(I,K,N),TH42(I,K,N),TH51(I,K,N),TH52(I,K,N),
TH61(I,K,N),TH62(I,K,N),TH71(K,N),TH72(K,N),TH73(K,N),TH74(K,N),TH75(K,N),TH76(K,N),
DEMAND1(K),DEMAND2(N),MAKESPAN;

*##############################################*
*#                  MODEL                     #*
*##############################################*
$include trial.gms

wv.fx(I,K,'N0')                = 0;
wv.fx('C1','4',N)              = 0;
wv.fx('C2','4',N)              = 0;
**wv.fx('C1','1','N2')          = 1;
wv.fx('1',K,N)$(ORD(N) <= 2) = 0;
wv.fx('2',K,N)$(ORD(N) <= 3) = 0;
wv.fx('3',K,N)$(ORD(N) <= 3) = 0;
```

```
wv.fx('C1',K,N)$(ORD(N) >= (CARD(N)-5)) = 0;
wv.fx('C2',K,N)$(ORD(N) >= (CARD(N)-5)) = 0;
wv.fx('1',K,N)$(ORD(N) >= (CARD(N)-4))  = 0;
wv.fx('2',K,N)$(ORD(N) >= (CARD(N)-3))  = 0;
wv.fx('3',K,N)$(ORD(N) >= (CARD(N)-3))  = 0;

B.fx(I,J,K,'NO')              = 0;
B.fx('2','2',K,N)$(ORD(N) <= 3) = 0;
B.fx('3','3',K,N)$(ORD(N) <= 3) = 0;
B.fx('C1','C1',K,N)$(ORD(N) >= (CARD(N)-5)) = 0;
B.fx('C2','C2',K,N)$(ORD(N) >= (CARD(N)-5)) = 0;
B.fx('1','1',K,N)$(ORD(N) >= (CARD(N)-4)) = 0;
B.fx('2','2',K,N)$(ORD(N) >= (CARD(N)-3)) = 0;
B.fx('3','3',K,N)$(ORD(N) >= (CARD(N)-3)) = 0;

*************** 1. ALLOCATION ***************
wv.fx('C2',K,N) =0;

A001(N).. SUM(kk, wv('C1',kk,N) )                                          =L= 1;
A101(N).. SUM(kk, wv('1',kk,N) )                                           =L= 1;
A102(N).. SUM(kk, wv('2',kk,N) )                                           =L= 1;
A103(N).. SUM(kk, wv('3',kk,N) )                                           =L= 1;
A104(N).. SUM(kk, wv('4_1',kk,N) + wv('4_2',kk,N) + wv('4_3',kk,N) + wv('4_4',kk,N) )   =L= 1;

A105(N).. SUM(kk, wv('5_1',kk,N) + wv('5_2',kk,N)) =L= 1;
A106(N).. SUM(kk, wv('6_1',kk,N) + wv('6_2',kk,N)) =L= 1;
A107(N).. SUM(kk, wv('5_1',kk,N) + wv('6_1',kk,N)) =L= 1;
A108(N).. SUM(kk, wv('5_2',kk,N) + wv('6_2',kk,N)) =L= 1;

A109(K,N)..  wv('7_1',K,N) + wv('7_2',K,N) + wv('7_3',K,N)          =E= 3*yv('7',N);
A110(N)..  SUM(kk,wv('7_1',kk,N) + wv('7_2',kk,N) + wv('7_3',kk,N))  =L= 3;
A111(N)..  yv('7',N) =L= 1;

* Ensure K is consistent for processing of a batch
A201(K,N)..      wv('1',K,N)   =L= wv('2',K,N+1);
A202(K,N)..      wv('1',K,N)   =L= wv('3',K,N+1);
** only one of the tasks 4_2 and 4_1 can happen directly after
A203(K,N)..      wv('1',K,N)   =L= SUM(UNIT4(ii),wv(ii,K,N+1));
A204(K,N)..      wv('1',K,N)   =L= SUM(UNIT4(ii),wv(ii,K,N+2));
A205(K,N)..      wv('1',K,N)   =L= SUM(UNIT4(ii),wv(ii,K,N+3));
A206(K,N)..      wv('1',K,N)   =L= SUM(UNIT4(ii),wv(ii,K,N+4));
A207(K,N)..      wv('2',K,N)   =L= wv('5_1',K,N+1) + wv('6_1',K,N+1);
A208(K,N)..      wv('3',K,N)   =L= wv('5_2',K,N+1) + wv('6_2',K,N+1);
A209(K,N)..      wv('2',K,N)   =L= wv('7_1',K,N+4);
A210(K,N)..      wv('3',K,N)   =L= wv('7_2',K,N+4);
A211(K,N)..      wv('5_1',K,N) + wv('6_1',K,N)  =L= wv('7_1',K,N+3);
A212(K,N)..      wv('5_2',K,N) + wv('6_2',K,N)  =L= wv('7_2',K,N+3);

A301(K,N).. B('1','1',K,N) =L= (1/SPLIT('F1',K)) *  B('2','2',K,N+1);
A302(K,N).. B('1','1',K,N) =L= (1/SPLIT('F2',K)) *  B('3','3',K,N+1);
A303(K,N).. B('1','1',K,N) =L= (1/SPLIT('F3',K)) * (B('4_1','4',K,N+1) + B('4_1','4',K,N+2) +
                                                    B('4_1','4',K,N+3)+ B('4_1','4',K,N+4));
A304(K,N).. B('1','1',K,N) =L= (1/SPLIT('F4',K)) * (B('4_2','4',K,N+1) + B('4_2','4',K,N+2) +
                                                    B('4_2','4',K,N+3)+ B('4_2','4',K,N+4));
A305(K,N).. B('2','2',K,N) =L= (1/SPLIT2('Fi',K))* (B('4_3','4',K,N+1) + B('4_3','4',K,N+2) +
                                                    B('4_3','4',K,N+3) );
A306(K,N).. B('3','3',K,N) =L= (1/SPLIT2('Fii',K))*(B('4_4','4',K,N+1)+ B('4_4','4',K,N+2) +
                                                    B('4_4','4',K,N+3) );
A307(K,N).. B('2','2',K,N) =L= (1/(1-SPLIT2('Fi',K))) *(B('5_1','5',K,N+1) + B('6_1','6',K,N+1));
A308(K,N).. B('3','3',K,N) =L= (1/(1-SPLIT2('Fii',K)))*(B('5_2','5',K,N+1) + B('6_2','6',K,N+1));
A309(K,N).. B('5_1','5',K,N) + B('6_1','6',K,N) =L= B('7_1','7',K,N+3);
A310(K,N).. B('5_2','5',K,N) + B('6_2','6',K,N) =L= B('7_2','7',K,N+3);
A311(K,N).. B('1','1',K,N) =E= SUM(UNIT7(ii),B(ii,'7',K,N+5));
A312(K)$(ORD(K) <= 3).. SUM(nn,B('C1','C1',K,nn) + B('C2','C2',K,nn)) =G= STIN(K);
A313(K)$(ORD(K) <= 3).. SUM(nn,B('1','1',K,nn)) =G= F('1')*SUM(nn,B('C1','C1',K,nn) +
                                                    F('2')*B('C2','C2',K,nn));

A401(I,K,N)..              wv(I,K,N)$(ORD(N) = CARD(N)) =E= 0;

A500(N)$( ((ORD(N) > 1) AND (ORD(N) <= (CARD(N)-12)) ) )..
                                                    SUM(kk,wv('C1',kk,N))                    =G= 1;
A501(N)$( (MOD(ORD(N),4) = 3) AND (ORD(N)/2 <= (CARD(K)+9)) AND (ORD(N) > 2) )..
                                                    SUM(kk,wv('1',kk,N))                     =G= 1;
A502(N)$( (MOD(ORD(N),4) = 0) AND (ORD(N)/2 <= (CARD(K)+10)) AND (ORD(N) > 3) )..
                                                    SUM(kk,wv('2',kk,N)   + wv('3',kk,N))    =G= 2;
```

```
A503(N)$( (MOD(ORD(N),4) = 0) AND (ORD(N)/2 <= (CARD(K)+10)) AND (ORD(N) > 3) )..
                                        SUM((UNIT4(ii),kk),wv(ii,kk,N))        =G= 1;
A504(N)$( (MOD(ORD(N),4) = 1) AND (ORD(N)/2 <= (CARD(K)+11)) AND (ORD(N) > 3) )..
                                        SUM((UNIT4(ii),kk),wv(ii,kk,N))        =G= 1;
A505(N)$( (MOD(ORD(N),4) = 2) AND (ORD(N)/2 <= (CARD(K)+11)) AND (ORD(N) > 3) )..
                                        SUM((UNIT4(ii),kk),wv(ii,kk,N))        =G= 1;
A506(N)$( (MOD(ORD(N),4) = 3) AND (ORD(N)/2 <= (CARD(K)+12)) AND (ORD(N) > 3) )..
                                        SUM((UNIT4(ii),kk),wv(ii,kk,N))        =G= 1;
A507(N)$( (MOD(ORD(N),4) = 0) AND (ORD(N)/3 <= (CARD(K)+12)) AND (ORD(N) > 5) )..
                                        SUM(kk,yv('7',kk,N))                   =G= 1;
A508(N)$( (MOD(ORD(N),4) = 1) AND (ORD(N)/2 <= (CARD(K)+11)) AND (ORD(N) > 3) )..
                                        SUM((UNIT56(ii),kk),wv(ii,kk,N))       =G= 2;

*************** 2. UNIT CAPACITY ***************
CCi(K,N)..  B('C1','C1',K,N)  =L= 100*wv('C1',K,N);
CCii(K,N).. B('C2','C2',K,N)  =L= 100*wv('C2',K,N);
CC01(K,N)..  B('1','1',K,N)    =L= 50*wv('1',K,N);
CC02(K,N)..  B('2','2',K,N)    =L= 40*wv('2',K,N);
CC03(K,N)..  B('3','3',K,N)    =L= 40*wv('3',K,N);
CC04(K,N)..  B('4_1','4',K,N)  =L= 40*wv('4_1',K,N);
CC05(K,N)..  B('4_2','4',K,N)  =L= 40*wv('4_2',K,N);
CC06(K,N)..  B('4_3','4',K,N)  =L= 30*wv('4_3',K,N);
CC07(K,N)..  B('4_4','4',K,N)  =L= 30*wv('4_4',K,N);
CC08(K,N)..  B('5_1','5',K,N)  =L= 40*wv('5_1',K,N);
CC09(K,N)..  B('5_2','5',K,N)  =L= 40*wv('5_2',K,N);
CC10(K,N)..  B('6_1','6',K,N)  =L= 40*wv('6_1',K,N);
CC11(K,N)..  B('6_2','6',K,N)  =L= 40*wv('6_2',K,N);
CC12(K,N)..  B('7_1','7',K,N)  =L=50*wv('7_1',K,N);
CC13(K,N)..  B('7_2','7',K,N)  =L=50*wv('7_2',K,N);
CC14(K,N)..  B('7_3','7',K,N)  =L=50*wv('7_3',K,N);

CC001(K,N) .. B('C1','C1',K,N)  =G= 10*wv('C1',K,N);
CC002(K,N)..  B('C2','C2',K,N)  =G= 10*wv('C2',K,N);
CC101(K,N)..  B('1','1',K,N)    =G= 10*wv('1',K,N);
CC102(K,N)..  B('2','2',K,N)    =G= 1*wv('2',K,N);
CC103(K,N)..  B('3','3',K,N)    =G= 1*wv('3',K,N);
CC104(K,N)..  B('4_1','4',K,N)  =G= 1*wv('4_1',K,N);
CC105(K,N)..  B('4_2','4',K,N)  =G= 1*wv('4_2',K,N);
CC106(K,N)..  B('4_3','4',K,N)  =G= 0.01*wv('4_3',K,N);
CC107(K,N)..  B('4_4','4',K,N)  =G= 0.01*wv('4_4',K,N);
CC108(K,N)..  B('5_1','5',K,N)  =G= 0.01*wv('5_1',K,N);
CC109(K,N)..  B('5_2','5',K,N)  =G= 0.01*wv('5_2',K,N);
CC110(K,N)..  B('6_1','6',K,N)  =G= 0.01*wv('6_1',K,N);
CC111(K,N)..  B('6_2','6',K,N)  =G= 0.01*wv('6_2',K,N);
CC112(K,N)..  B('7_1','7',K,N)  =G= 1*wv('7_1',K,N);
CC113(K,N)..  B('7_2','7',K,N)  =G= 1*wv('7_2',K,N);
CC114(K,N)..  B('7_3','7',K,N)  =G= 1*wv('7_3',K,N);

*************** 3. STORAGE CAPACITY ***************

SC1(S,N)$(ORD(S) <= CARD(K))..ST(S,N)       =L= 250;
SC2(K,N)..  ST('S2',N)    =L= 50*(1 - wv('1',K,N));
SC3(K,N)..  ST('S3',N)    =L= 50*(1 - wv('1',K,N));
SC4(K,N)..  ST('S4',N)    =L= 50*(1 - wv('1',K,N));
SC5(K,N)..  ST('S5',N)    =L= 50*(1 - wv('2',K,N));
SC6(K,N)..  ST('S6',N)    =L= 50*(1 - wv('3',K,N));
SC7(N)..    ST('S7',N)    =L= 50;
SC8(N)..    ST('S8',N)    =L= 50;
SC9(N)..    ST('S9',N)    =L= 50;
SC10(N)..   ST('S10',N)   =L= 50;
SC11(N)..   ST('S11',N)   =L= 50;
SC12(N)..   ST('S12',N)   =L= 50;
SC13(N)..   ST('S13',N)   =L= 50;

*************** 4. MASS BALANCE ***************

MB0(K,N)$((ORD(K) <= 3))..Si(K,N) =E= (STIN(K) - B('C1','C1',K,N))$(ORD(N) EQ 1)
                                    + (Si(K,N-1) - B('C1','C1',K,N))$(ORD(N) GT 1);

MB1(S,K,N)$((ORD(S) <= CARD(K)) AND (ORD(S) = ORD (K)))..
                                    ST(S,N) =E= (STIN1(K) - B('1','1',K,N) )$(ORD(N) EQ 1)
                    + (ST(S,N-1) + FR(N)*B('C1','C1',K,N-1) - B('1','1',K,N))$(ORD(N) GT 1);

MB2(N)..  ST('S2',N) =E= (0 + SUM(kk, - B('2','2',kk,N) ))$(ORD(N) EQ 1)
                        + (ST('S2',N-1) + SUM(kk, SPLIT('F1',kk)*B('1','1',kk,N-1)
```

```
                        - B('2','2',kk,N) ))$(ORD(N) GT 1);

MB3(N)..  ST('S3',N) =E= (0 + SUM(kk, B('3','3',kk,N) ))$(ORD(N) EQ 1)
                         + (ST('S3',N-1) + SUM(kk, SPLIT('F2',kk)*B('1','1',kk,N-1)
                         - B('3','3',kk,N) ))$(ORD(N) GT 1);

MB4(N)..  ST('S4',N) =E= (0 - SUM(kk, B('4_1','4',kk,N) ))$(ORD(N) EQ 1)
                         + (ST('S4',N-1) + SUM(kk, SPLIT('F3',kk)*B('1','1',kk,N-1)
                         - B('4_1','4',kk,N) ))$(ORD(N) GT 1);

MB5(N)..  ST('S5',N) =E= (0 - SUM(kk, B('4_2','4',kk,N) ))$(ORD(N) EQ 1)
                         + (ST('S5',N-1) + SUM(kk, SPLIT('F4',kk)*B('1','1',kk,N-1)
                         - B('4_2','4',kk,N) ))$(ORD(N) GT 1);

MB6(N)..  ST('S6',N) =E= (0 - SUM(kk, B('5_1','5',kk,N) - B('6_1','6',kk,N)))$(ORD(N) EQ 1)
                         + (ST('S6',N-1) + SUM(kk, (1-SPLIT2('Fi',kk))*B('2','2',kk,N-1)
                         - B('5_1','5',kk,N) - B('6_1','6',kk,N) ))$(ORD(N) GT 1);

MB7(N)..  ST('S7',N) =E= (0 - SUM(kk, B('4_3','4',kk,N) ))$(ORD(N) EQ 1)
                         + (ST('S7',N-1) + SUM(kk, SPLIT2('Fi',kk)*B('2','2',kk,N-1)
                         - B('4_3','4',kk,N) ))$(ORD(N) GT 1);

MB8(N)..  ST('S8',N) =E= (0 - SUM(kk, B('5_2','5',kk,N) - B('6_2','6',kk,N) ))$(ORD(N) EQ 1)
                         + (ST('S8',N-1) + SUM(kk, (1-SPLIT2('Fii',kk))*B('3','3',kk,N-1)
                         - B('5_2','5',kk,N) - B('6_2','6',kk,N) ))$(ORD(N) GT 1);

MB9(N)..  ST('S9',N) =E= (0 - SUM(kk, B('4_4','4',kk,N) ))$(ORD(N) EQ 1)
                         + (ST('S9',N-1) + SUM(kk, SPLIT2('Fii',kk)*B('3','3',kk,N-1)
                         - B('4_4','4',kk,N) ))$(ORD(N) GT 1);

MB10(N)..  ST('S10',N) =E= (0 - SUM(kk, B('7_1','7',kk,N) ))$(ORD(N) EQ 1)
                          + (ST('S10',N-1) + SUM(kk, B('5_1','5',kk,N-1)  + B('6_1','6',kk,N-1))
                          - SUM(kk,B('7_1','7',kk,N)) )$(ORD(N) GT 1);

MB11(N)..  ST('S11',N) =E= (0 - SUM(kk, B('7_2','7',kk,N) ))$(ORD(N) EQ 1)
                          + (ST('S11',N-1) + SUM(kk, B('5_2','5',kk,N-1) + B('6_2','6',kk,N-1))
                          - SUM(kk,B('7_2','7',kk,N)) )$(ORD(N) GT 1);


MB12(N)..  ST('S12',N) =E= (0 - SUM(kk, B('7_3','7',kk,N) ))$(ORD(N) EQ 1)
                          + (ST('S12',N-1) + SUM((UNIT4(ii),kk), B(ii,'4',kk,N-1))
                          - SUM(kk,B('7_3','7',kk,N)) )$(ORD(N) GT 1);


MB20(N)..  ST('S20',N) =E= (0 - SUM(kk,D('S20',kk,N) ))$(ORD(N) EQ 1)
                          + (ST('S20',N-1) + SUM((UNIT7(ii),kk), B(ii,'7',kk,N-1))
                          - SUM(kk,D('S20',kk,N) ))$(ORD(N) GT 1);

************** 5. DURATION CONSTRAINTS ***************

DC(I,J,K,N).. Ts(I,J,K,N) =G= 0;

DCi(K,N)..   Tf('C1','C1',K,N)  =E= Ts('C1','C1',K,N) + B('C1','C1',K,N)*R.L(N);
DCii(K,N).. Tf('C2','C2',K,N)  =E= Ts('C2','C2',K,N) + B('C2','C2',K,N);
DCO1(K,N).. Tf('1','1',K,N)    =E= Ts('1','1',K,N)    + 20*wv('1',K,N)    + B('1','1',K,N)*3;
DCO2(K,N).. Tf('2','2',K,N)    =E= Ts('2','2',K,N)    + 10*wv('2',K,N)    + B('2','2',K,N)*15;
DCO3(K,N).. Tf('3','3',K,N)    =E= Ts('3','3',K,N)    + 10*wv('3',K,N)    + B('3','3',K,N)*13;
DCO4(K,N).. Tf('4_1','4',K,N)  =E= Ts('4_1','4',K,N) + 15*wv('4_1',K,N) + B('4_1','4',K,N)*8;
DCO5(K,N).. Tf('4_2','4',K,N)  =E= Ts('4_2','4',K,N) + 10*wv('4_2',K,N) + B('4_2','4',K,N)*8;
DCO6(K,N).. Tf('4_3','4',K,N)  =E= Ts('4_3','4',K,N) +  5*wv('4_3',K,N) + B('4_3','4',K,N)*9;
DCO7(K,N).. Tf('4_4','4',K,N)  =E= Ts('4_4','4',K,N) +  5*wv('4_4',K,N) + B('4_4','4',K,N)*9;

DCO8(K,N).. Tf('5_1','5',K,N)  =E= Ts('5_1','5',K,N) + 5*wv('5_1',K,N) + B('5_1','5',K,N)*10;
DCO9(K,N).. Tf('5_2','5',K,N)  =E= Ts('5_2','5',K,N) + 5*wv('5_2',K,N) + B('5_2','5',K,N)*7;
DC10(K,N).. Tf('6_1','6',K,N)  =E= Ts('6_1','6',K,N) + 5*wv('6_1',K,N) + B('6_1','6',K,N)*10;
DC11(K,N).. Tf('6_2','6',K,N)  =E= Ts('6_2','6',K,N) + 5*wv('6_2',K,N) + B('6_2','6',K,N)*7;

DC12(K,N).. Tf('7_1','7',K,N)  =E= Ts('7_1','7',K,N) + 150*wv('7_1',K,N);
DC13(K,N).. Tf('7_2','7',K,N)  =E= Ts('7_2','7',K,N) + 150*wv('7_2',K,N);
DC14(K,N).. Tf('7_3','7',K,N)  =E= Ts('7_3','7',K,N) + 150*wv('7_3',K,N);

************** 6.1 SAME TASK SAME UNIT ***************

STSU01(K,N+1).. Ts('C1','C1',K,N+1)  =G= Tf('C1','C1',K,N) - H*(1 - wv('C1',K,N));
STSU02(K,N+1).. Ts('C1','C1',K,N+1)  =G= Ts('C1','C1',K,N);
STSU03(K,N+1).. Tf('C1','C1',K,N+1)  =G= Tf('C1','C1',K,N);
```

```
STSU04(K,N+1).. Ts('C2','C2',K,N+1)  =G= Tf('C2','C2',K,N) - H*(1 - wv('C2',K,N));
STSU05(K,N+1).. Ts('C2','C2',K,N+1)  =G= Ts('C2','C2',K,N);
STSU06(K,N+1).. Tf('C2','C2',K,N+1)  =G= Tf('C2','C2',K,N);

STSU11(K,N+1).. Ts('1','1',K,N+1)  =G= Tf('1','1',K,N) - H*(1 - wv('1',K,N));
STSU12(K,N+1).. Ts('1','1',K,N+1)  =G= Ts('1','1',K,N);
STSU13(K,N+1).. Tf('1','1',K,N+1)  =G= Tf('1','1',K,N);

STSU21(K,N+1).. Ts('2','2',K,N+1)  =G= Tf('2','2',K,N) - H*(1 - wv('2',K,N));
STSU22(K,N+1).. Ts('2','2',K,N+1)  =G= Ts('2','2',K,N);
STSU23(K,N+1).. Tf('2','2',K,N+1)  =G= Tf('2','2',K,N);

STSU31(K,N+1).. Ts('3','3',K,N+1)  =G= Tf('3','3',K,N) - H*(1 - wv('3',K,N));
STSU32(K,N+1).. Ts('3','3',K,N+1)  =G= Ts('3','3',K,N);
STSU33(K,N+1).. Tf('3','3',K,N+1)  =G= Tf('3','3',K,N);

STSU41(K,N+1).. Ts('4_1','4',K,N+1) =G= Tf('4_1','4',K,N) - H*(1 - wv('4_1',K,N));
STSU42(K,N+1).. Ts('4_1','4',K,N+1) =G= Ts('4_1','4',K,N);
STSU43(K,N+1).. Tf('4_1','4',K,N+1) =G= Tf('4_1','4',K,N);

STSU44(K,N+1).. Ts('4_2','4',K,N+1) =G= Tf('4_2','4',K,N) - H*(1 - wv('4_2',K,N));
STSU45(K,N+1).. Ts('4_2','4',K,N+1) =G= Ts('4_2','4',K,N);
STSU46(K,N+1).. Tf('4_2','4',K,N+1) =G= Tf('4_2','4',K,N);

STSU47(K,N+1).. Ts('4_3','4',K,N+1) =G= Tf('4_3','4',K,N) - H*(1 - wv('4_3',K,N));
STSU48(K,N+1).. Ts('4_3','4',K,N+1) =G= Ts('4_3','4',K,N);
STSU49(K,N+1).. Tf('4_3','4',K,N+1) =G= Tf('4_3','4',K,N);

STSU4a(K,N+1).. Ts('4_4','4',K,N+1) =G= Tf('4_4','4',K,N) - H*(1 - wv('4_4',K,N));
STSU4b(K,N+1).. Ts('4_4','4',K,N+1) =G= Ts('4_4','4',K,N);
STSU4c(K,N+1).. Tf('4_4','4',K,N+1) =G= Tf('4_4','4',K,N);

STSU51(K,N+1).. Ts('5_1','5',K,N+1) =G= Tf('5_1','5',K,N) - H*(1 - wv('5_1',K,N));
STSU52(K,N+1).. Ts('5_1','5',K,N+1) =G= Ts('5_1','5',K,N);
STSU53(K,N+1).. Tf('5_1','5',K,N+1) =G= Tf('5_1','5',K,N);

STSU54(K,N+1).. Ts('5_2','5',K,N+1) =G= Tf('5_2','5',K,N) - H*(1 - wv('5_2',K,N));
STSU55(K,N+1).. Ts('5_2','5',K,N+1) =G= Ts('5_2','5',K,N);
STSU56(K,N+1).. Tf('5_2','5',K,N+1) =G= Tf('5_2','5',K,N);

STSU61(K,N+1).. Ts('6_1','6',K,N+1) =G= Tf('6_1','6',K,N) - H*(1 - wv('6_1',K,N));
STSU62(K,N+1).. Ts('6_1','6',K,N+1) =G= Ts('6_1','6',K,N);
STSU63(K,N+1).. Tf('6_1','6',K,N+1) =G= Tf('6_1','6',K,N);

STSU64(K,N+1).. Ts('6_2','6',K,N+1) =G= Tf('6_2','6',K,N) - H*(1 - wv('6_2',K,N));
STSU65(K,N+1).. Ts('6_2','6',K,N+1) =G= Ts('6_2','6',K,N);
STSU66(K,N+1).. Tf('6_2','6',K,N+1) =G= Tf('6_2','6',K,N);

STSU71(K,N+1).. Ts('7_1','7',K,N+1) =G= Tf('7_1','7',K,N) - H*(1 - wv('7_1',K,N));
STSU72(K,N+1).. Ts('7_1','7',K,N+1) =G= Ts('7_1','7',K,N);
STSU73(K,N+1).. Tf('7_1','7',K,N+1) =G= Tf('7_1','7',K,N);

STSU74(K,N+1).. Ts('7_2','7',K,N+1) =G= Tf('7_2','7',K,N) - H*(1 - wv('7_2',K,N));
STSU75(K,N+1).. Ts('7_2','7',K,N+1) =G= Ts('7_2','7',K,N);
STSU76(K,N+1).. Tf('7_2','7',K,N+1) =G= Tf('7_2','7',K,N);

STSU77(K,N+1).. Ts('7_3','7',K,N+1) =G= Tf('7_3','7',K,N) - H*(1 - wv('7_3',K,N));
STSU78(K,N+1).. Ts('7_3','7',K,N+1) =G= Ts('7_3','7',K,N);
STSU79(K,N+1).. Tf('7_3','7',K,N+1) =G= Tf('7_3','7',K,N);

*************** 6.2 DIFFERENT TASK SAME UNIT ***************

DTSU01(K,N+1)$(NOT(ORD(K) = 1)).. Ts('C1','C1','1',N+1) =G= Tf('C1','C1',K,N)
                                                            - H*(1 - wv('C1',K,N));
DTSU02(K,N+1)$(NOT(ORD(K) = 2)).. Ts('C1','C1','2',N+1) =G= Tf('C1','C1',K,N)
                                                            - H*(1 - wv('C1',K,N));
DTSU03(K,N+1)$(NOT(ORD(K) = 3)).. Ts('C1','C1','3',N+1) =G= Tf('C1','C1',K,N)
                                                            - H*(1 - wv('C1',K,N));
DTSU04(K,N+1)$(NOT(ORD(K) = 4)).. Ts('C1','C1','4',N+1) =G= Tf('C1','C1',K,N)
                                                            - H*(1 - wv('C1',K,N));

DTSU05(K,N+1)$(NOT(ORD(K) = 1)).. Ts('C2','C2','1',N+1) =G= Tf('C2','C2',K,N)
                                                            - H*(1 - wv('C2',K,N));
DTSU06(K,N+1)$(NOT(ORD(K) = 2)).. Ts('C2','C2','2',N+1) =G= Tf('C2','C2',K,N)
                                                            - H*(1 - wv('C2',K,N));
```

```
DTSU07(K,N+1)$(NOT(ORD(K) = 3)).. Ts('C2','C2','3',N+1) =G= Tf('C2','C2',K,N)
                                                          - H*(1 - wv('C2',K,N));
DTSU08(K,N+1)$(NOT(ORD(K) = 4)).. Ts('C2','C2','4',N+1) =G= Tf('C2','C2',K,N)
                                                          - H*(1 - wv('C2',K,N));

DTSU11(K,N+1)$(NOT(ORD(K) = 1)).. Ts('1','1','1',N+1) =G= Tf('1','1',K,N) - H*(1 - wv('1',K,N));
DTSU12(K,N+1)$(NOT(ORD(K) = 2)).. Ts('1','1','2',N+1) =G= Tf('1','1',K,N) - H*(1 - wv('1',K,N));
DTSU13(K,N+1)$(NOT(ORD(K) = 3)).. Ts('1','1','3',N+1) =G= Tf('1','1',K,N) - H*(1 - wv('1',K,N));
DTSU14(K,N+1)$(NOT(ORD(K) = 4)).. Ts('1','1','4',N+1) =G= Tf('1','1',K,N) - H*(1 - wv('1',K,N));

DTSU21(K,N+1)$(NOT(ORD(K) = 1)).. Ts('2','2','1',N+1) =G= Tf('2','2',K,N) - H*(1 - wv('2',K,N));
DTSU22(K,N+1)$(NOT(ORD(K) = 2)).. Ts('2','2','2',N+1) =G= Tf('2','2',K,N) - H*(1 - wv('2',K,N));
DTSU23(K,N+1)$(NOT(ORD(K) = 3)).. Ts('2','2','3',N+1) =G= Tf('2','2',K,N) - H*(1 - wv('2',K,N));
DTSU24(K,N+1)$(NOT(ORD(K) = 4)).. Ts('2','2','4',N+1) =G= Tf('2','2',K,N) - H*(1 - wv('2',K,N));

DTSU31(K,N+1)$(NOT(ORD(K) = 1)).. Ts('3','3','1',N+1) =G= Tf('3','3',K,N) - H*(1 - wv('3',K,N));
DTSU32(K,N+1)$(NOT(ORD(K) = 2)).. Ts('3','3','2',N+1) =G= Tf('3','3',K,N) - H*(1 - wv('3',K,N));
DTSU33(K,N+1)$(NOT(ORD(K) = 3)).. Ts('3','3','3',N+1) =G= Tf('3','3',K,N) - H*(1 - wv('3',K,N));
DTSU34(K,N+1)$(NOT(ORD(K) = 4)).. Ts('3','3','4',N+1) =G= Tf('3','3',K,N) - H*(1 - wv('3',K,N));

DTSU41(K,N+1)$(NOT(ORD(K) = 1)).. Ts('4_1','4','1',N+1) =G= Tf('4_1','4',K,N)
                                                          - H*(1 - wv('4_1',K,N));
DTSU42(K,N+1)$(NOT(ORD(K) = 2)).. Ts('4_1','4','2',N+1) =G= Tf('4_1','4',K,N)
                                                          - H*(1 - wv('4_1',K,N));
DTSU43(K,N+1)$(NOT(ORD(K) = 3)).. Ts('4_1','4','3',N+1) =G= Tf('4_1','4',K,N)
                                                          - H*(1 - wv('4_1',K,N));
DTSU47(K,N+1)$(NOT(ORD(K) = 4)).. Ts('4_1','4','4',N+1) =G= Tf('4_1','4',K,N)
                                                          - H*(1 - wv('4_1',K,N));

DTSU44(K,N+1)$(NOT(ORD(K) = 1)).. Ts('4_2','4','1',N+1) =G= Tf('4_2','4',K,N)
                                                          - H*(1 - wv('4_2',K,N));
DTSU45(K,N+1)$(NOT(ORD(K) = 2)).. Ts('4_2','4','2',N+1) =G= Tf('4_2','4',K,N)
                                                          - H*(1 - wv('4_2',K,N));
DTSU46(K,N+1)$(NOT(ORD(K) = 3)).. Ts('4_2','4','3',N+1) =G= Tf('4_2','4',K,N)
                                                          - H*(1 - wv('4_2',K,N));
DTSU48(K,N+1)$(NOT(ORD(K) = 4)).. Ts('4_2','4','4',N+1) =G= Tf('4_2','4',K,N)
                                                          - H*(1 - wv('4_2',K,N));

DTSU401(K,N+1)$(NOT(ORD(K) = 1)).. Ts('4_3','4','1',N+1) =G= Tf('4_3','4',K,N)
                                                          - H*(1 - wv('4_3',K,N));
DTSU402(K,N+1)$(NOT(ORD(K) = 2)).. Ts('4_3','4','2',N+1) =G= Tf('4_3','4',K,N)
                                                          - H*(1 - wv('4_3',K,N));
DTSU403(K,N+1)$(NOT(ORD(K) = 3)).. Ts('4_3','4','3',N+1) =G= Tf('4_3','4',K,N)
                                                          - H*(1 - wv('4_3',K,N));
DTSU404(K,N+1)$(NOT(ORD(K) = 4)).. Ts('4_3','4','4',N+1) =G= Tf('4_3','4',K,N)
                                                          - H*(1 - wv('4_3',K,N));

DTSU405(K,N+1)$(NOT(ORD(K) = 1)).. Ts('4_4','4','1',N+1) =G= Tf('4_4','4',K,N)
                                                          - H*(1 - wv('4_4',K,N));
DTSU406(K,N+1)$(NOT(ORD(K) = 2)).. Ts('4_4','4','2',N+1) =G= Tf('4_4','4',K,N)
                                                          - H*(1 - wv('4_4',K,N));
DTSU407(K,N+1)$(NOT(ORD(K) = 3)).. Ts('4_4','4','3',N+1) =G= Tf('4_4','4',K,N)
                                                          - H*(1 - wv('4_4',K,N));
DTSU408(K,N+1)$(NOT(ORD(K) = 4)).. Ts('4_4','4','4',N+1) =G= Tf('4_4','4',K,N)
                                                          - H*(1 - wv('4_4',K,N));

DTSU51(K,N+1)$(NOT(ORD(K) = 1)).. Ts('5_1','5','1',N+1) =G= Tf('5_1','5',K,N)
                                                          - H*(1 - wv('5_1',K,N));
DTSU52(K,N+1)$(NOT(ORD(K) = 2)).. Ts('5_1','5','2',N+1) =G= Tf('5_1','5',K,N)
                                                          - H*(1 - wv('5_1',K,N));
DTSU53(K,N+1)$(NOT(ORD(K) = 3)).. Ts('5_1','5','3',N+1) =G= Tf('5_1','5',K,N)
                                                          - H*(1 - wv('5_1',K,N));
DTSU54(K,N+1)$(NOT(ORD(K) = 4)).. Ts('5_1','5','4',N+1) =G= Tf('5_1','5',K,N)
                                                          - H*(1 - wv('5_1',K,N));

DTSU55(K,N+1)$(NOT(ORD(K) = 1)).. Ts('5_2','5','1',N+1) =G= Tf('5_2','5',K,N)
                                                          - H*(1 - wv('5_2',K,N));
DTSU56(K,N+1)$(NOT(ORD(K) = 2)).. Ts('5_2','5','2',N+1) =G= Tf('5_2','5',K,N)
                                                          - H*(1 - wv('5_2',K,N));
DTSU57(K,N+1)$(NOT(ORD(K) = 3)).. Ts('5_2','5','3',N+1) =G= Tf('5_2','5',K,N)
                                                          - H*(1 - wv('5_2',K,N));
DTSU58(K,N+1)$(NOT(ORD(K) = 4)).. Ts('5_2','5','4',N+1) =G= Tf('5_2','5',K,N)
                                                          - H*(1 - wv('5_2',K,N));

DTSU61(K,N+1)$(NOT(ORD(K) = 1)).. Ts('6_1','6','1',N+1) =G= Tf('6_1','6',K,N)
```

```
                                                                - H*(1 - wv('6_1',K,N));
DTSU62(K,N+1)$(NOT(ORD(K) = 2)).. Ts('6_1','6','2',N+1) =G= Tf('6_1','6',K,N)
                                                                - H*(1 - wv('6_1',K,N));
DTSU63(K,N+1)$(NOT(ORD(K) = 3)).. Ts('6_1','6','3',N+1) =G= Tf('6_1','6',K,N)
                                                                - H*(1 - wv('6_1',K,N));
DTSU64(K,N+1)$(NOT(ORD(K) = 4)).. Ts('6_1','6','4',N+1) =G= Tf('6_1','6',K,N)
                                                                - H*(1 - wv('6_1',K,N));

DTSU65(K,N+1)$(NOT(ORD(K) = 1)).. Ts('6_2','6','1',N+1) =G= Tf('6_2','6',K,N)
                                                                - H*(1 - wv('6_2',K,N));
DTSU66(K,N+1)$(NOT(ORD(K) = 2)).. Ts('6_2','6','2',N+1) =G= Tf('6_2','6',K,N)
                                                                - H*(1 - wv('6_2',K,N));
DTSU67(K,N+1)$(NOT(ORD(K) = 3)).. Ts('6_2','6','3',N+1) =G= Tf('6_2','6',K,N)
                                                                - H*(1 - wv('6_2',K,N));
DTSU68(K,N+1)$(NOT(ORD(K) = 4)).. Ts('6_2','6','4',N+1) =G= Tf('6_2','6',K,N)
                                                                - H*(1 - wv('6_2',K,N));

DTSU71(K,N+1)$(NOT(ORD(K) = 1)).. Ts('7_1','7','1',N+1) =G= Tf('7_1','7',K,N)
                                                                - H*(1 - wv('7_1',K,N));
DTSU72(K,N+1)$(NOT(ORD(K) = 2)).. Ts('7_1','7','2',N+1) =G= Tf('7_1','7',K,N)
                                                                - H*(1 - wv('7_1',K,N));
DTSU73(K,N+1)$(NOT(ORD(K) = 3)).. Ts('7_1','7','3',N+1) =G= Tf('7_1','7',K,N)
                                                                - H*(1 - wv('7_1',K,N));
DTSU74(K,N+1)$(NOT(ORD(K) = 4)).. Ts('7_1','7','4',N+1) =G= Tf('7_1','7',K,N)
                                                                - H*(1 - wv('7_1',K,N));

DTSU75(K,N+1)$(NOT(ORD(K) = 1)).. Ts('7_2','7','1',N+1) =G= Tf('7_2','7',K,N)
                                                                - H*(1 - wv('7_2',K,N));
DTSU76(K,N+1)$(NOT(ORD(K) = 2)).. Ts('7_2','7','2',N+1) =G= Tf('7_2','7',K,N)
                                                                - H*(1 - wv('7_2',K,N));
DTSU77(K,N+1)$(NOT(ORD(K) = 3)).. Ts('7_2','7','3',N+1) =G= Tf('7_2','7',K,N)
                                                                - H*(1 - wv('7_2',K,N));
DTSU78(K,N+1)$(NOT(ORD(K) = 4)).. Ts('7_2','7','4',N+1) =G= Tf('7_2','7',K,N)
                                                                - H*(1 - wv('7_2',K,N));

DTSU79(K,N+1)$(NOT(ORD(K) = 1)).. Ts('7_3','7','1',N+1) =G= Tf('7_3','7',K,N)
                                                                - H*(1 - wv('7_3',K,N));
DTSU710(K,N+1)$(NOT(ORD(K) = 2)).. Ts('7_3','7','2',N+1) =G= Tf('7_3','7',K,N)
                                                                - H*(1 - wv('7_3',K,N));
DTSU711(K,N+1)$(NOT(ORD(K) = 3)).. Ts('7_3','7','3',N+1) =G= Tf('7_3','7',K,N)
                                                                - H*(1 - wv('7_3',K,N));
DTSU712(K,N+1)$(NOT(ORD(K) = 4)).. Ts('7_3','7','4',N+1) =G= Tf('7_3','7',K,N)
                                                                - H*(1 - wv('7_3',K,N));

* extra constraints added for approximation of continuous plant as 2 units
DTSU001(K,N+1).. Ts('C1','C1','1',N+1)  =G= Tf('C2','C2',K,N) - H*(1 - wv('C2',K,N));
DTSU002(K,N+1).. Ts('C1','C1','2',N+1)  =G= Tf('C2','C2',K,N) - H*(1 - wv('C2',K,N));
DTSU003(K,N+1).. Ts('C1','C1','3',N+1)  =G= Tf('C2','C2',K,N) - H*(1 - wv('C2',K,N));
DTSU004(K,N+1).. Ts('C1','C1','4',N+1)  =G= Tf('C2','C2',K,N) - H*(1 - wv('C2',K,N));
DTSU005(K,N+1).. Ts('C2','C2','1',N+1)  =G= Tf('C1','C1',K,N) - H*(1 - wv('C1',K,N));
DTSU006(K,N+1).. Ts('C2','C2','2',N+1)  =G= Tf('C1','C1',K,N) - H*(1 - wv('C1',K,N));
DTSU007(K,N+1).. Ts('C2','C2','3',N+1)  =G= Tf('C1','C1',K,N) - H*(1 - wv('C1',K,N));
DTSU008(K,N+1).. Ts('C2','C2','4',N+1)  =G= Tf('C1','C1',K,N) - H*(1 - wv('C1',K,N));
*****************************************************************************

DTSU411(I,K,N+1)$((ORD(I)>= 6) AND (ORD(I) <= 9)).. Ts('4_1','4','1',N+1) =G= Tf(I,'4',K,N)
                                                                - H*(1 - wv(I,K,N));
DTSU412(I,K,N+1)$((ORD(I)>= 6) AND (ORD(I) <= 9)).. Ts('4_1','4','2',N+1) =G= Tf(I,'4',K,N)
                                                                - H*(1 - wv(I,K,N));
DTSU413(I,K,N+1)$((ORD(I)>= 6) AND (ORD(I) <= 9)).. Ts('4_1','4','3',N+1) =G= Tf(I,'4',K,N)
                                                                - H*(1 - wv(I,K,N));
DTSU414(I,K,N+1)$((ORD(I)>= 6) AND (ORD(I) <= 9)).. Ts('4_1','4','4',N+1) =G= Tf(I,'4',K,N)
                                                                - H*(1 - wv(I,K,N));

DTSU415(I,K,N+1)$((ORD(I)>= 6) AND (ORD(I) <= 9)).. Ts('4_2','4','1',N+1) =G= Tf(I,'4',K,N)
                                                                - H*(1 - wv(I,K,N));
DTSU416(I,K,N+1)$((ORD(I)>= 6) AND (ORD(I) <= 9)).. Ts('4_2','4','2',N+1) =G= Tf(I,'4',K,N)
                                                                - H*(1 - wv(I,K,N));
DTSU417(I,K,N+1)$((ORD(I)>= 6) AND (ORD(I) <= 9)).. Ts('4_2','4','3',N+1) =G= Tf(I,'4',K,N)
                                                                - H*(1 - wv(I,K,N));
DTSU418(I,K,N+1)$((ORD(I)>= 6) AND (ORD(I) <= 9)).. Ts('4_2','4','4',N+1) =G= Tf(I,'4',K,N)
                                                                - H*(1 - wv(I,K,N));

DTSU419(I,K,N+1)$((ORD(I)>= 6) AND (ORD(I) <= 9)).. Ts('4_3','4','1',N+1) =G= Tf(I,'4',K,N)
                                                                - H*(1 - wv(I,K,N));
```

```
DTSU420(I,K,N+1)$((ORD(I)>= 6) AND (ORD(I) <= 9)).. Ts('4_3','4','2',N+1) =G= Tf(I,'4',K,N)
                                                                              - H*(1 - wv(I,K,N));
DTSU422(I,K,N+1)$((ORD(I)>= 6) AND (ORD(I) <= 9)).. Ts('4_3','4','3',N+1) =G= Tf(I,'4',K,N)
                                                                              - H*(1 - wv(I,K,N));
DTSU423(I,K,N+1)$((ORD(I)>= 6) AND (ORD(I) <= 9)).. Ts('4_3','4','4',N+1) =G= Tf(I,'4',K,N)
                                                                              - H*(1 - wv(I,K,N));

DTSU424(I,K,N+1)$((ORD(I)>= 6) AND (ORD(I) <= 8)).. Ts('4_4','4','1',N+1) =G= Tf(I,'4',K,N)
                                                                              - H*(1 - wv(I,K,N));
DTSU425(I,K,N+1)$((ORD(I)>= 6) AND (ORD(I) <= 8)).. Ts('4_4','4','2',N+1) =G= Tf(I,'4',K,N)
                                                                              - H*(1 - wv(I,K,N));
DTSU426(I,K,N+1)$((ORD(I)>= 6) AND (ORD(I) <= 8)).. Ts('4_4','4','3',N+1) =G= Tf(I,'4',K,N)
                                                                              - H*(1 - wv(I,K,N));
DTSU427(I,K,N+1)$((ORD(I)>= 6) AND (ORD(I) <= 8)).. Ts('4_4','4','4',N+1) =G= Tf(I,'4',K,N)
                                                                              - H*(1 - wv(I,K,N));

DTSU511(K,N+1)..      Ts('5_1','5','1',N+1)  =G= Tf('5_2','5',K,N) - H*(1 - wv('5_2',K,N));
DTSU512(K,N+1)..      Ts('5_1','5','2',N+1)  =G= Tf('5_2','5',K,N) - H*(1 - wv('5_2',K,N));
DTSU513(K,N+1)..      Ts('5_1','5','3',N+1)  =G= Tf('5_2','5',K,N) - H*(1 - wv('5_2',K,N));
DTSU514(K,N+1)..      Ts('5_1','5','4',N+1)  =G= Tf('5_2','5',K,N) - H*(1 - wv('5_2',K,N));

DTSU515(K,N+1)..      Ts('5_2','5','1',N+1)  =G= Tf('5_1','5',K,N) - H*(1 - wv('5_1',K,N));
DTSU516(K,N+1)..      Ts('5_2','5','2',N+1)  =G= Tf('5_1','5',K,N) - H*(1 - wv('5_1',K,N));
DTSU517(K,N+1)..      Ts('5_2','5','3',N+1)  =G= Tf('5_1','5',K,N) - H*(1 - wv('5_1',K,N));
DTSU518(K,N+1)..      Ts('5_2','5','4',N+1)  =G= Tf('5_1','5',K,N) - H*(1 - wv('5_1',K,N));

DTSU611(K,N+1)..      Ts('6_1','6','1',N+1)  =G= Tf('6_2','6',K,N) - H*(1 - wv('6_2',K,N));
DTSU612(K,N+1)..      Ts('6_1','6','2',N+1)  =G= Tf('6_2','6',K,N) - H*(1 - wv('6_2',K,N));
DTSU613(K,N+1)..      Ts('6_1','6','3',N+1)  =G= Tf('6_2','6',K,N) - H*(1 - wv('6_2',K,N));
DTSU614(K,N+1)..      Ts('6_1','6','4',N+1)  =G= Tf('6_2','6',K,N) - H*(1 - wv('6_2',K,N));

DTSU615(K,N+1)..      Ts('6_2','6','1',N+1)  =G= Tf('6_1','6',K,N) - H*(1 - wv('6_1',K,N));
DTSU616(K,N+1)..      Ts('6_2','6','2',N+1)  =G= Tf('6_1','6',K,N) - H*(1 - wv('6_1',K,N));
DTSU617(K,N+1)..      Ts('6_2','6','3',N+1)  =G= Tf('6_1','6',K,N) - H*(1 - wv('6_1',K,N));
DTSU618(K,N+1)..      Ts('6_2','6','4',N+1)  =G= Tf('6_1','6',K,N) - H*(1 - wv('6_1',K,N));

*************** 6.3 DIFFERENT TASK DIFFERENT UNIT ***************

DTDU11(K,N+1).. Ts('1','1',K,N+1)    =G= Tf('C1','C1',K,N) - H*(1 - wv('C1',K,N));
DTDU12(K,N+1).. Ts('1','1',K,N+1)    =G= Tf('C2','C2',K,N) - H*(1 - wv('C2',K,N));

DTDU21(K,N+1).. Ts('2','2',K,N+1)    =G= Tf('1','1',K,N) - H*(1 - wv('1',K,N));

DTDU31(K,N+1).. Ts('3','3',K,N+1)    =G= Tf('1','1',K,N) - H*(1 - wv('1',K,N));

DTDU411(K,N+1).. Ts('4_1','4',K,N+1)  =G= Tf('1','1',K,N) - H*(1 - wv('1',K,N));
DTDU421(K,N+1).. Ts('4_2','4',K,N+1)  =G= Tf('1','1',K,N) - H*(1 - wv('1',K,N));
DTDU431(K,N+1).. Ts('4_3','4',K,N+1)  =G= Tf('2','2',K,N) - H*(1 - wv('2',K,N));
DTDU441(K,N+1).. Ts('4_4','4',K,N+1)  =G= Tf('3','3',K,N) - H*(1 - wv('3',K,N));

DTDU511(K,N+1).. Ts('5_1','5',K,N+1)  =G= Tf('2','2',K,N) - H*(1 - wv('2',K,N));
DTDU521(K,N+1).. Ts('6_1','6',K,N+1)  =G= Tf('2','2',K,N) - H*(1 - wv('2',K,N));

DTDU611(K,N+1).. Ts('5_2','5',K,N+1)  =G= Tf('3','3',K,N) - H*(1 - wv('3',K,N));
DTDU621(K,N+1).. Ts('6_2','6',K,N+1)  =G= Tf('3','3',K,N) - H*(1 - wv('3',K,N));

DTDU7a1(K,N+1).. Ts('7_1','7',K,N+1)  =G= Tf('5_1','5',K,N) - H*(1 - wv('5_1',K,N));
DTDU7a2(K,N+1).. Ts('7_1','7',K,N+1)  =G= Tf('6_1','6',K,N) - H*(1 - wv('6_1',K,N));

DTDU7b1(K,N+1).. Ts('7_2','7',K,N+1)  =G= Tf('5_2','5',K,N) - H*(1 - wv('5_2',K,N));
DTDU7b2(K,N+1).. Ts('7_2','7',K,N+1)  =G= Tf('6_2','6',K,N) - H*(1 - wv('6_2',K,N));

DTDU7c1(K,N+1).. Ts('7_3','7',K,N+1)  =G= Tf('4_1','4',K,N) - H*(1 - wv('4_1',K,N));
DTDU7c2(K,N+1).. Ts('7_3','7',K,N+1)  =G= Tf('4_2','4',K,N) - H*(1 - wv('4_2',K,N));
DTDU7c3(K,N+1).. Ts('7_3','7',K,N+1)  =G= Tf('4_3','4',K,N) - H*(1 - wv('4_3',K,N));
DTDU7c4(K,N+1).. Ts('7_3','7',K,N+1)  =G= Tf('4_4','4',K,N) - H*(1 - wv('4_4',K,N));

************* No Mixing *********************************

DTDU201(K,N+1).. Ts('2','2','1',N)  =L= Tf('1','1',K,N+1) + H*(1 - wv('1',K,N+1));
DTDU202(K,N+1).. Ts('2','2','2',N)  =L= Tf('1','1',K,N+1) + H*(1 - wv('1',K,N+1));
DTDU203(K,N+1).. Ts('2','2','3',N)  =L= Tf('1','1',K,N+1) + H*(1 - wv('1',K,N+1));
DTDU204(K,N+1).. Ts('2','2','4',N)  =L= Tf('1','1',K,N+1) + H*(1 - wv('1',K,N+1));

DTDU301(K,N+1).. Ts('3','3','1',N)  =L= Tf('1','1',K,N+1) + H*(1 - wv('1',K,N+1));
```

```
DTDU302(K,N+1).. Ts('3','3','2',N)  =L= Tf('1','1',K,N+1) + H*(1 - wv('1',K,N+1));
DTDU303(K,N+1).. Ts('3','3','3',N)  =L= Tf('1','1',K,N+1) + H*(1 - wv('1',K,N+1));
DTDU304(K,N+1).. Ts('3','3','4',N)  =L= Tf('1','1',K,N+1) + H*(1 - wv('1',K,N+1));

DTDU401(UNIT4a(I),K,N+1).. Ts(I,'4','1',N)  =L= Tf('1','1',K,N) + H*(1 - wv('1',K,N));
DTDU402(UNIT4a(I),K,N+1).. Ts(I,'4','2',N)  =L= Tf('1','1',K,N) + H*(1 - wv('1',K,N));
DTDU403(UNIT4a(I),K,N+1).. Ts(I,'4','3',N)  =L= Tf('1','1',K,N) + H*(1 - wv('1',K,N));
DTDU404(UNIT4a(I),K,N+1).. Ts(I,'4','4',N)  =L= Tf('1','1',K,N) + H*(1 - wv('1',K,N));

DTDU405(UNIT4a(I),K,N+1).. Ts(I,'4','1',N)  =L= Tf('1','1',K,N+1) + H*(1 - wv('1',K,N+1));
DTDU406(UNIT4a(I),K,N+1).. Ts(I,'4','2',N)  =L= Tf('1','1',K,N+1) + H*(1 - wv('1',K,N+1));
DTDU407(UNIT4a(I),K,N+1).. Ts(I,'4','3',N)  =L= Tf('1','1',K,N+1) + H*(1 - wv('1',K,N+1));
DTDU408(UNIT4a(I),K,N+1).. Ts(I,'4','4',N)  =L= Tf('1','1',K,N+1) + H*(1 - wv('1',K,N+1));

DTDU4a1(K,N+1).. Ts('4_3','4','1',N)  =L= Tf('2','2',K,N+1) + H*(1 - wv('2',K,N+1));
DTDU4a2(K,N+1).. Ts('4_3','4','2',N)  =L= Tf('2','2',K,N+1) + H*(1 - wv('2',K,N+1));
DTDU4a3(K,N+1).. Ts('4_3','4','3',N)  =L= Tf('2','2',K,N+1) + H*(1 - wv('2',K,N+1));
DTDU4a4(K,N+1).. Ts('4_3','4','4',N)  =L= Tf('2','2',K,N+1) + H*(1 - wv('2',K,N+1));

DTDU4a5(K,N+1).. Ts('4_4','4','1',N)  =L= Tf('3','3',K,N+1) + H*(1 - wv('3',K,N+1));
DTDU4a6(K,N+1).. Ts('4_4','4','2',N)  =L= Tf('3','3',K,N+1) + H*(1 - wv('3',K,N+1));
DTDU4a7(K,N+1).. Ts('4_4','4','3',N)  =L= Tf('3','3',K,N+1) + H*(1 - wv('3',K,N+1));
DTDU4a8(K,N+1).. Ts('4_4','4','4',N)  =L= Tf('3','3',K,N+1) + H*(1 - wv('3',K,N+1));

DTDU501(K,N+1).. Ts('5_1','5','1',N)  =L= Tf('2','2',K,N+1) + H*(1 - wv('2',K,N+1));
DTDU502(K,N+1).. Ts('5_1','5','2',N)  =L= Tf('2','2',K,N+1) + H*(1 - wv('2',K,N+1));
DTDU503(K,N+1).. Ts('5_1','5','3',N)  =L= Tf('2','2',K,N+1) + H*(1 - wv('2',K,N+1));
DTDU504(K,N+1).. Ts('5_1','5','4',N)  =L= Tf('2','2',K,N+1) + H*(1 - wv('2',K,N+1));

DTDU505(K,N+1).. Ts('5_2','5','1',N)  =L= Tf('3','3',K,N+1) + H*(1 - wv('3',K,N+1));
DTDU506(K,N+1).. Ts('5_2','5','2',N)  =L= Tf('3','3',K,N+1) + H*(1 - wv('3',K,N+1));
DTDU507(K,N+1).. Ts('5_2','5','3',N)  =L= Tf('3','3',K,N+1) + H*(1 - wv('3',K,N+1));
DTDU508(K,N+1).. Ts('5_2','5','4',N)  =L= Tf('3','3',K,N+1) + H*(1 - wv('3',K,N+1));

DTDU601(K,N+1).. Ts('6_1','6','1',N)  =L= Tf('2','2',K,N+1) + H*(1 - wv('2',K,N+1));
DTDU602(K,N+1).. Ts('6_1','6','2',N)  =L= Tf('2','2',K,N+1) + H*(1 - wv('2',K,N+1));
DTDU603(K,N+1).. Ts('6_1','6','3',N)  =L= Tf('2','2',K,N+1) + H*(1 - wv('2',K,N+1));
DTDU604(K,N+1).. Ts('6_1','6','4',N)  =L= Tf('2','2',K,N+1) + H*(1 - wv('2',K,N+1));

DTDU605(K,N+1).. Ts('6_2','6','1',N)  =L= Tf('3','3',K,N+1) + H*(1 - wv('3',K,N+1));
DTDU606(K,N+1).. Ts('6_2','6','2',N)  =L= Tf('3','3',K,N+1) + H*(1 - wv('3',K,N+1));
DTDU607(K,N+1).. Ts('6_2','6','3',N)  =L= Tf('3','3',K,N+1) + H*(1 - wv('3',K,N+1));
DTDU608(K,N+1).. Ts('6_2','6','4',N)  =L= Tf('3','3',K,N+1) + H*(1 - wv('3',K,N+1));

DTDU701(K,N+1).. Ts('7_1','7','1',N)  =L= Tf('5_1','5',K,N+1) + H*(1 - wv('5_1',K,N+1));
DTDU702(K,N+1).. Ts('7_1','7','2',N)  =L= Tf('5_1','5',K,N+1) + H*(1 - wv('5_1',K,N+1));
DTDU703(K,N+1).. Ts('7_1','7','3',N)  =L= Tf('5_1','5',K,N+1) + H*(1 - wv('5_1',K,N+1));
DTDU704(K,N+1).. Ts('7_1','7','4',N)  =L= Tf('5_1','5',K,N+1) + H*(1 - wv('5_1',K,N+1));
DTDU705(K,N+1).. Ts('7_1','7','1',N)  =L= Tf('6_1','6',K,N+1) + H*(1 - wv('6_1',K,N+1));
DTDU706(K,N+1).. Ts('7_1','7','2',N)  =L= Tf('6_1','6',K,N+1) + H*(1 - wv('6_1',K,N+1));
DTDU707(K,N+1).. Ts('7_1','7','3',N)  =L= Tf('6_1','6',K,N+1) + H*(1 - wv('6_1',K,N+1));
DTDU708(K,N+1).. Ts('7_1','7','4',N)  =L= Tf('6_1','6',K,N+1) + H*(1 - wv('6_1',K,N+1));

DTDU709(K,N+1).. Ts('7_2','7','1',N)  =L= Tf('5_2','5',K,N+1) + H*(1 - wv('5_2',K,N+1));
DTDU710(K,N+1).. Ts('7_2','7','2',N)  =L= Tf('5_2','5',K,N+1) + H*(1 - wv('5_2',K,N+1));
DTDU711(K,N+1).. Ts('7_2','7','3',N)  =L= Tf('5_2','5',K,N+1) + H*(1 - wv('5_2',K,N+1));
DTDU712(K,N+1).. Ts('7_2','7','4',N)  =L= Tf('5_2','5',K,N+1) + H*(1 - wv('5_2',K,N+1));
DTDU713(K,N+1).. Ts('7_2','7','1',N)  =L= Tf('6_2','6',K,N+1) + H*(1 - wv('6_2',K,N+1));
DTDU714(K,N+1).. Ts('7_2','7','2',N)  =L= Tf('6_2','6',K,N+1) + H*(1 - wv('6_2',K,N+1));
DTDU715(K,N+1).. Ts('7_2','7','3',N)  =L= Tf('6_2','6',K,N+1) + H*(1 - wv('6_2',K,N+1));
DTDU716(K,N+1).. Ts('7_2','7','4',N)  =L= Tf('6_2','6',K,N+1) + H*(1 - wv('6_2',K,N+1));

DTDU717(UNIT4(I),K,N+1).. Ts('7_3','7','1',N)  =L= Tf(I,'4',K,N+1) + H*(1 - wv(I,K,N+1));
DTDU718(UNIT4(I),K,N+1).. Ts('7_3','7','2',N)  =L= Tf(I,'4',K,N+1) + H*(1 - wv(I,K,N+1));
DTDU719(UNIT4(I),K,N+1).. Ts('7_3','7','3',N)  =L= Tf(I,'4',K,N+1) + H*(1 - wv(I,K,N+1));
DTDU720(UNIT4(I),K,N+1).. Ts('7_3','7','4',N)  =L= Tf(I,'4',K,N+1) + H*(1 - wv(I,K,N+1));

*************** 6.4 COMPLETION OF PREVIOUS TASKS ***************

CPT1(I,J,K,N+1)$((ORD(I) EQ ORD(J)) AND (ORD(I) LE 5))..
            Ts(I,J,K,N+1) =G= SUM((kk,Ite(N,nn)),Tf(I,J,kk,nn) - Ts(I,J,kk,nn));

CPT2(I,K,N+1)$((ORD(I)>= 6) AND (ORD(I) <= 9))..
           Ts(I,'4',K,N+1) =G= SUM((UNIT4(ii),kk,Ite(N,nn)),Tf(ii,'4',kk,nn) - Ts(ii,'4',kk,nn));
```

```
CPT3(I,K,N+1)$((ORD(I)>= 10) AND (ORD(I) <= 11))..
           Ts(I,'5',K,N+1) =G= SUM((UNIT5(ii),kk,Ite(N,nn)),Tf(ii,'5',kk,nn) - Ts(ii,'5',kk,nn));

CPT4(I,K,N+1)$((ORD(I)>= 12) AND (ORD(I) <= 13))..
           Ts(I,'6',K,N+1) =G= SUM((UNIT6(ii),kk,Ite(N,nn)),Tf(ii,'6',kk,nn) - Ts(ii,'6',kk,nn));

CPT5(K,N+1)..Ts('7_1','7',K,N+1) =G= SUM((kk,Ite(N,nn)),Tf('7_1','7',kk,nn)-Ts('7_1','7',kk,nn));
CPT6(K,N+1)..Ts('7_2','7',K,N+1) =G= SUM((kk,Ite(N,nn)),Tf('7_1','7',kk,nn)-Ts('7_1','7',kk,nn));
CPT7(K,N+1)..Ts('7_3','7',K,N+1) =G= SUM((kk,Ite(N,nn)),Tf('7_1','7',kk,nn)-Ts('7_1','7',kk,nn));
CPT8(K,N+1)..Ts('7_3','7',K,N+1) =E= Ts('7_1','7',K,N+1);
CPT9(K,N+1)..Ts('7_2','7',K,N+1) =E= Ts('7_1','7',K,N+1);

*************** 7. TIME HORIZON ***************
TH01(K,N).. Ts('C1','C1',K,N)   =L= H;
TH02(K,N).. Tf('C1','C1',K,N)   =L= H;
TH03(K,N).. Ts('C2','C2',K,N)   =L= H;
TH04(K,N).. Tf('C2','C2',K,N)   =L= H;
TH11(K,N).. Ts('1','1',K,N)   =L= H;
TH12(K,N).. Tf('1','1',K,N)   =L= H;
TH21(K,N).. Ts('2','2',K,N)   =L= H;
TH22(K,N).. Tf('2','2',K,N)   =L= H;
TH31(K,N).. Ts('3','3',K,N)   =L= H;
TH32(K,N).. Tf('3','3',K,N)   =L= H;

TH41(UNIT4(I),K,N).. Ts(I,'4',K,N) =L= H;
TH42(UNIT4(I),K,N).. Tf(I,'4',K,N) =L= H;
TH51(UNIT5(I),K,N).. Ts(I,'5',K,N) =L= H;
TH52(UNIT5(I),K,N).. Tf(I,'5',K,N) =L= H;
TH61(UNIT6(I),K,N).. Ts(I,'6',K,N) =L= H;
TH62(UNIT6(I),K,N).. Tf(I,'6',K,N) =L= H;

TH71(K,N).. Ts('7_1','7',K,N) =L= H;
TH72(K,N).. Ts('7_2','7',K,N) =L= H;
TH73(K,N).. Ts('7_3','7',K,N) =L= H;
TH74(K,N).. Tf('7_1','7',K,N) =L= H;
TH75(K,N).. Tf('7_2','7',K,N) =L= H;
TH76(K,N).. Ts('7_3','7',K,N) =L= H;

DEMAND1(K).. SUM(nn, D('S20',K,nn))  =G=  SUM((nn), FR(nn)*B('C1','C1',K,nn) ) + STIN1(K);
DEMAND2(N)$(ORD(N)=CARD(N)).. SUM(kk, Si(kk,N))  =L=  0;

*************** 8. OBJECTIVE FUNCTION ***************

MAKESPAN(N)$(ORD(N) = CARD(N))..
              Z =G= SUM(kk,Tf('7_1','7',kk,N) + Tf('7_2','7',kk,N) + Tf('7_3','7',kk,N));

*************************************
*               SOLVER              *
*************************************
MODEL FISH /ALL/
OPTION optcr =0.05;
OPTION MIP = CPLEX;
OPTION reslim = 720000;
OPTION iterlim = 100000000;
FISH.workspace = 500;
FISH.OptFile = 1;

FILE randsimulation1 /randsimulation1.xls/;
FILE randsimulation2 /randsimulation2.xls/;
FILE randsimulation3 /randsimulation3.xls/;
FILE randsimulation4 /randsimulation4.xls/;
FILE randsimulation5 /randsimulation5.xls/;

alias(N,Ni);
Set trial /trial1*trial5/;
Variable rates(trial,Ni), fractions(trial,Ni), objfun(trial);
    LOOP(trial,
        LOOP(Ni,
        p(Ni) = uniform(0,1);
        R.L(Ni) = a('1') + (a('2') - a('1'))*p(Ni);
        FR(Ni) = bi*R.L(Ni) + c;
        rates.L(trial,Ni) = R.L(Ni);
        fractions.L(trial,Ni) = FR(Ni));
SOLVE FISH USING MIP MINIMIZING Z;
```

```
* WRITE OUTPUT TO FILE *
*======================*
*print control (5=comma delimited)
randsimulation1.pc=5;
randsimulation2.pc=5;
randsimulation3.pc=5;
randsimulation4.pc=5;
randsimulation5.pc=5;

*whether to append to file or not
*randsimulation1.ap=0;
*randsimulation2.ap=0;

* number of decimals displayed
randsimulation1.nd=0;
randsimulation2.nd=0;
randsimulation3.nd=0;
randsimulation4.nd=0;
randsimulation5.nd=0;

*numeric format (1= rounded to fit fields)
randsimulation1.nr=1;
randsimulation2.nr=1;
randsimulation3.nr=1;
randsimulation4.nr=1;
randsimulation5.nr=1;

    IF( ORD(trial) = 1,
        PUT randsimulation1;
    ELSEIF( ORD(trial) = 2),
        PUT randsimulation2;
    ELSEIF( ORD(trial) = 3),
        PUT randsimulation3;
    ELSEIF( ORD(trial) = 4),
        PUT randsimulation4;
    ELSEIF( ORD(trial) = 5),
        PUT randsimulation5;
        );

    LOOP((K,N,I,J),
    If (Tf.L(I,J,K,N) gt (Ts.L(I,J,K,N)+1),
            PUT I.tl PUT J.tl PUT K.tl PUT N.tl PUT Ts.L(I,J,K,N), PUT Tf.L(I,J,K,N), PUT / ));

PUTCLOSE;

        objfun.L(trial) = Z.L);

************** DISPLAY ***************
Total.L(K)   =  SUM((ss,nn),D.L(ss,K,nn));
Total1.L(K)  =  SUM((ii,nn),B.L(ii,'1',K,nn));
Total2.L(K)  =  SUM((ii,nn),B.L(ii,'2',K,nn));
Total3.L(K)  =  SUM((ii,nn),B.L(ii,'3',K,nn));
Total4.L(K)  =  SUM((ii,nn),B.L(ii,'4',K,nn));
Total5.L(K)  =  SUM((ii,nn),B.L(ii,'7',K,nn));

OPTION wv:1:2:1;
OPTION ST:2:1:1;
OPTION Ts:2:3:1;
OPTION Tf:2:3:1;
OPTION  B:2:3:1;
OPTION  D:2:2:1;
OPTION Si:2:1:1;
OPTION objfun:2:0:1;

DISPLAY
*wv.L,*yv.L,*ST.L,Si.L,B.L,D.L,Ts.L,Tf.L,STIN,Total.L,Total1.L,Total2.L,Total3.L,Total4.L,Total5.L,
rates.L,fractions.L,objfun.L;
```

A.2 FORTRAN Code

The aim of this code is to generate random data which is used to determine the sequencing of material in the batch plant.

FORTRAN RANDOM DATA

```
C Fri Sep 1 21:07:30 SAST 2000
C Aims of this code
C 1. LEARN how program in FORTRAN!!!! hence the inefficient code
C 2. Generate input data for random simulation
C 3. Output Data is edited using sed which constructs
C       the GAMS input files

       IMPLICIT REAL*8 (a-h,o-z)
       dimension icount(7), icount2(4), icountu4(4), icountu41(4)
       integer*2 yomama(350,8), batch, Order, Orderu4, Unit4, Unit5

            LOOPS = 50
            IN = 7*LOOPS
C change yomama(?,7) to the same value as IN

C seed obtained from CPU clock and replaced using 'sed'
C prior to compilation
            IseedXXX

C initialize the matrix
        DO 5 I = 1,IN
            DO 6 J= 1,8
             yomama(I,J) = 0
6           CONTINUE
5           CONTINUE

C outside loop - number of times to run
      DO 15 J = 1, LOOPS

        DO 20 I = 1, 7
            icount(I) = 0
20      CONTINUE

        DO 25 K = 1, 4
            icount2(K) = 0
25      CONTINUE

        DO 26 L = 1, 4
            icountu4(L) = 0
            icountu41(L) = 0
26      CONTINUE

          R = RAND(Iseed)
          Iseed = Iseed + 1

C determine the order ie between 1 and 7
30        Order = NInt(R*6) + 1

            if  ( Order .eq. 1)then
                icount(1) = icount(1) + 1

                elseif ( Order .eq. 2)then
                icount(2) = icount(2) + 1

                elseif ( Order .eq. 3)then
                icount(3) = icount(3) + 1

                elseif ( Order .eq. 4)then
                icount(4) = icount(4) + 1

                elseif ( Order .eq. 5)then
                icount(5) = icount(5) + 1

                elseif ( Order .eq. 6)then
```

```
                icount(6) = icount(6) + 1

                elseif ( Order .eq. 7)then
                icount(7) = icount(7) + 1

             endif

            if ((icount(Order) .gt. 1))then
                R = RAND(Iseed)
                Iseed = Iseed + 1

                GOTO 30

            else
                icount3 = order + (J-1)*7
                yomama(icount3,1) = J

                yomama(icount3,2) = Order
C generate a random number to associate a source with the
C number between 1 and 7
                R = RAND(Iseed)
                Iseed = Iseed + 1

C randomly select a source
50              batch = NInt(R*3) + 1

                    if((icount2(1) .gt. 2) .and. (icount2(2) .gt. 2)
     &                 .and.(icount2(3) .gt. 1) .and.
     &                      (icount2(4) .gt. 2))then

                      GOTO 100

                    else
                      if  ( batch .eq. 1)then
                            icount2(1) = icount2(1) + 1

                                       if (icount2(1) .GT. 2)then
                                        R = RAND(Iseed)
                                        Iseed = Iseed + 1
                                        GOTO 50
                                       endif

                            elseif ( batch .eq. 2)then
                            icount2(2) = icount2(2) + 1

                                       if (icount2(2) .GT. 2)then
                                        R = RAND(Iseed)
                                        Iseed = Iseed + 1
                                        GOTO 50
                                       endif

                            elseif ( batch .eq. 3)then
                            icount2(3) = icount2(3) + 1

                                       if (icount2(3) .GT. 1)then
                                        R = RAND(Iseed)
                                        Iseed = Iseed + 1
                                        GOTO 50
                                       endif

                            elseif ( batch .eq. 4)then
                            icount2(4) = icount2(4) + 1

                                       if (icount2(4) .GT. 2)then
                                        R = RAND(Iseed)
                                        Iseed = Iseed + 1
                                        GOTO 50
                                       endif

                      endif
                            yomama(icount3,3) = batch

C determine the order for processing on unit 4
C tasks 4_1 and 4_2 are in the set unit4a
C one these tasks take place at the first event point after task 1
```

```
150                     Ra = RAND(Iseed)
                        Iseed = Iseed + 1

                        Unit4a = NInt(Ra) + 1
                      if(Unit4a .eq. 1)then
                            icountu41(1) = icountu41(1) + 1
                            yomama(icount3,4) = Unit4a

                      else
                            icountu41(2) = icountu41(2) + 1
                            yomama(icount3,4) = Unit4a

167                   endif

C determine the order ie from 2-4
120                 R = RAND(Iseed)
                    Iseed = Iseed + 1
                    Orderu4 = NInt(R*2) + 2

                    if(icountu4(2)*icountu4(3)*icountu4(4)
     &                     .gt. 0)then
                        goto 212
                    else

                    if ( Orderu4 .eq. 2)then
                        icountu4(2) = icountu4(2) + 1
                        if ((icountu4(Orderu4) .gt. 1))then
                            GOTO 120
                        endif

                    elseif ( Orderu4 .eq. 3)then
                        icountu4(3) = icountu4(3) + 1
                        if ((icountu4(Orderu4) .gt. 1))then
                            GOTO 120
                        endif

                    elseif ( Orderu4 .eq. 4)then
                        icountu4(4) = icountu4(4) + 1
                        if ((icountu4(Orderu4) .gt. 1))then
                            GOTO 120
                        endif

138                 endif
                    endif

                      if(Orderu4 .eq. 2)then
170                       R = RAND(Iseed)
                          Iseed = Iseed + 1
                          Unit4 = NInt(R*3) + 1

                          if(Unit4 .eq. 1)then
                              icountu41(1) = icountu41(1) + 1
                              if(icountu41(Unit4) .gt. 1)then
                              goto 170
                              endif

                          elseif(Unit4 .eq. 2)then
                              icountu41(2) = icountu41(2) + 1
                              if(icountu41(Unit4) .gt. 1)then
                              goto 170
                              endif

                          elseif(Unit4 .eq. 3)then
                              icountu41(3) = icountu41(3) + 1
                              if(icountu41(Unit4) .gt. 1)then
                              goto 170
                              endif

                          elseif(Unit4 .eq. 4)then
                              icountu41(4) = icountu41(4) + 1
                              if(icountu41(Unit4) .gt. 1)then
                              goto 170
                              endif

217                       endif
                              yomama(icount3,5) = Unit4
```

```
                    elseif(Orderu4 .eq. 3)then
175                      R = RAND(Iseed)
                         Iseed = Iseed + 1
                         Unit4 = NInt(R*3) + 1

                         if(Unit4 .eq. 1)then
                              icountu41(1) = icountu41(1) + 1
                              if(icountu41(Unit4) .gt. 1)then
                              goto 175
                              endif

                           elseif(Unit4 .eq. 2)then
                              icountu41(2) = icountu41(2) + 1
                              if(icountu41(Unit4) .gt. 1)then
                              goto 175
                              endif

                           elseif(Unit4 .eq. 3)then
                              icountu41(3) = icountu41(3) + 1
                              if(icountu41(Unit4) .gt. 1)then
                              goto 175
                              endif

                           elseif(Unit4 .eq. 4)then
                              icountu41(4) = icountu41(4) + 1
                              if(icountu41(Unit4) .gt. 1)then
                              goto 175
                              endif

229                      endif
                          yomama(icount3,6) = Unit4

                    elseif(Orderu4 .eq. 4)then
180                      R = RAND(Iseed)
                         Iseed = Iseed + 1
                         Unit4 = NInt(R*3) + 1

                         if(Unit4 .eq. 1)then
                              icountu41(1) = icountu41(1) + 1
                              if(icountu41(Unit4) .gt. 1)then
                              goto 180
                              endif

                           elseif(Unit4 .eq. 2)then
                              icountu41(2) = icountu41(2) + 1
                              if(icountu41(Unit4) .gt. 1)then
                              goto 180
                              endif

                           elseif(Unit4 .eq. 3)then
                              icountu41(3) = icountu41(3) + 1
                              if(icountu41(Unit4) .gt. 1)then
                              goto 180
                              endif

                           elseif(Unit4 .eq. 4)then
                              icountu41(4) = icountu41(4) + 1
                              if(icountu41(Unit4) .gt. 1)then
                              goto 180
                              endif

280                      endif
                          yomama(icount3,7) = Unit4

                          R = RAND(Iseed)
                          Iseed = Iseed + 1
                          Unit5 = NInt(R)+1
                          if(Unit5 .eq. 1)then
                              yomama(icount3,8) = Unit5
                          else
                             yomama(icount3,8) = Unit5
                          endif

163               endif
```

```
212               if (icountu4(2)*icountu4(3)*icountu4(4)
     &                        .eq. 0)then
                          goto 120
                  else
211                       DO 226 L = 1, 4
                             icountu4(L) = 0
                             icountu41(L) = 0
226                       CONTINUE
                  endif
100       endif

66     endif

               if (icount(1)*icount(2)*icount(3)*icount(4)*icount(5)*
     &           icount(6)*icount(7) .eq. 0)then
                   R= RAND(Iseed)
                   Iseed = Iseed + 1
                   goto 30
               endif

15     CONTINUE

               do 345 I=1,IN
                  IF (MOD(I-1,7) .eq. 0 ) write(*,*)
     &             '+++++++++++++++++++++++++++++++++++++++++++++++++'
               write(*,600) yomama(I,1), yomama(I,2), yomama(I,3),
     &                      yomama(I,4), yomama(I,5), yomama(I,6),
     &                      yomama(I,7),yomama(I,8)
 345           continue
600     format(8(i6))

        STOP
        END
```

A.3 Bash Script

The purpose of this script is to interprete that randomly generated data from the FROTRAN script, generate the GAMS input files, run the simulations and sort data.

Bash Script to run Random Scenarios

```
#!/bin/bash2
# Sort out files
# Tue Sep  5 10:39:15 SAST 2000
# usage: ./sort
# step 1 - get generic output filename:

# use sed to generate a random seed from the clock time and insert into fortran
./randomseed
echo ""
echo "########## RANDOM SEED GENERATED ##########"
# compile fortran code
g77 -o randsimm.C randsim1.f
echo ""
echo "##########    FORTRAN COMPILED   ##########"
echo ""
# run fortran code to generate random data
./randsimm.C > trial.txt

# cut the first field before the delimiter .
OUTFILE=`echo trial.txt | cut -f 1 -d .`

trial=0

# the number of sets within the script is 50
while [ $trial -lt 50 ] ; do

        # read array in as variables A-G from trial.txt
                A=(`sed -n $[trial*8+2]p  trial.txt`)
                B=(`sed -n $[trial*8+2+1]p trial.txt`)
                C=(`sed -n $[trial*8+2+2]p trial.txt`)
                D=(`sed -n $[trial*8+2+3]p trial.txt`)
                E=(`sed -n $[trial*8+2+4]p trial.txt`)
                F=(`sed -n $[trial*8+2+5]p trial.txt`)
                G=(`sed -n $[trial*8+2+6]p trial.txt`)

        # create gams input file
        # where inputT0.gms is the template
        echo "wv.fx('1' , '$A[2]','N2')          = 1;" > $OUTFILE.tmp
        echo "wv.fx('2' , '$A[2]','N3')          = 1;" >>$OUTFILE.tmp
        echo "wv.fx('3' , '$A[2]','N3')          = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$A[3]','$A[2]','N3')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$A[4]','$A[2]','N4')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$A[5]','$A[2]','N5')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$A[6]','$A[2]','N6')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('5_$A[7]','$A[2]','N4')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('7_1','$A[2]','N7')          = 1;" >>$OUTFILE.tmp
        echo "                                     " >>$OUTFILE.tmp
        echo "wv.fx('1' , '$B[2]','N6')          = 1;" >>$OUTFILE.tmp
        echo "wv.fx('2' , '$B[2]','N7')          = 1;" >>$OUTFILE.tmp
        echo "wv.fx('3' , '$B[2]','N7')          = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$B[3]','$B[2]','N7')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$B[4]','$B[2]','N8')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$B[5]','$B[2]','N9')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$B[6]','$B[2]','N10')     = 1;" >>$OUTFILE.tmp
        echo "wv.fx('5_$B[7]','$B[2]','N8')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('7_1','$B[2]','N11')         = 1;" >>$OUTFILE.tmp
        echo "                                     " >>$OUTFILE.tmp
        echo "wv.fx('1' , '$C[2]','N10')         = 1;" >>$OUTFILE.tmp
        echo "wv.fx('2' , '$C[2]','N11')         = 1;" >>$OUTFILE.tmp
        echo "wv.fx('3' , '$C[2]','N11')         = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$C[3]','$C[2]','N11')     = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$C[4]','$C[2]','N12')     = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$C[5]','$C[2]','N13')     = 1;" >>$OUTFILE.tmp
```

```
        echo "wv.fx('4_$C[6]','$C[2]','N14')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('5_$C[7]','$C[2]','N12')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('7_1','$C[2]','N15')          = 1;" >>$OUTFILE.tmp
        echo "                                        " >>$OUTFILE.tmp
        echo "wv.fx('1' , '$D[2]','N14')          = 1;" >>$OUTFILE.tmp
        echo "wv.fx('2' , '$D[2]','N15')          = 1;" >>$OUTFILE.tmp
        echo "wv.fx('3' , '$D[2]','N15')          = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$D[3]','$D[2]','N15')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$D[4]','$D[2]','N16')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$D[5]','$D[2]','N17')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$D[6]','$D[2]','N18')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('5_$D[7]','$D[2]','N16')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('7_1','$D[2]','N19')          = 1;" >>$OUTFILE.tmp
        echo "                                        " >>$OUTFILE.tmp
        echo "wv.fx('1' , '$E[2]','N18')          = 1;" >>$OUTFILE.tmp
        echo "wv.fx('2' , '$E[2]','N19')          = 1;" >>$OUTFILE.tmp
        echo "wv.fx('3' , '$E[2]','N19')          = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$E[3]','$E[2]','N19')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$E[4]','$E[2]','N20')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$E[5]','$E[2]','N21')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$E[6]','$E[2]','N22')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('5_$E[7]','$E[2]','N20')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('7_1','$E[2]','N23')          = 1;" >>$OUTFILE.tmp
        echo "                                        " >>$OUTFILE.tmp
        echo "wv.fx('1' , '$F[2]','N22')          = 1;" >>$OUTFILE.tmp
        echo "wv.fx('2' , '$F[2]','N23')          = 1;" >>$OUTFILE.tmp
        echo "wv.fx('3' , '$F[2]','N23')          = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$F[3]','$F[2]','N23')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$F[4]','$F[2]','N24')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$F[5]','$F[2]','N25')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$F[6]','$F[2]','N26')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('5_$F[7]','$F[2]','N24')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('7_1','$F[2]','N27')          = 1;" >>$OUTFILE.tmp
        echo "                                        " >>$OUTFILE.tmp
        echo "wv.fx('1' , '$G[2]','N26')          = 1;" >>$OUTFILE.tmp
        echo "wv.fx('2' , '$G[2]','N27')          = 1;" >>$OUTFILE.tmp
        echo "wv.fx('3' , '$G[2]','N27')          = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$G[3]','$G[2]','N27')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$G[4]','$G[2]','N28')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$G[5]','$G[2]','N29')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('4_$G[6]','$G[2]','N30')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('5_$G[7]','$G[2]','N28')      = 1;" >>$OUTFILE.tmp
        echo "wv.fx('7_1','$G[2]','N31')          = 1;" >>$OUTFILE.tmp

# copy input file to temp file
    mv $OUTFILE.tmp $OUTFILE.gms
    cp $OUTFILE.gms input/$OUTFILE.$trial
# run gams
    echo "********** STARTING TRIAL $trial OF 49 **********"
    echo ""
    seed2=`date +%H%m%S`
    sed -e "s/XXYYZZ/ $seed2 /g" randsimulation.gms.template > randsimulation.tmp
    mv randsimulation.tmp randsimulation.gms
    echo "### NEW SEED FOR CONTINUOUS PLANT IS $seed2 ###"

   time gams randsimulation.gms ps=9999 pw=255
#  save results to file
    mv randsimulation.lst results/$OUTFILE$trial.lst
    echo "complete"
    testing=1
    while [ $testing -lt 6 ] ; do
     echo "***** moving file randsimulation$testing.xls *****"
   mv randsimulation$testing.xls results/data/randsimulation_$OUTFILE$trial_run$testing.xls
     echo "complete"
     echo ""
     testing=$[testing+1]
    done
trial=$[trial+1]
done
sleep 10s
# bash script to get the objective function from result files
./data
# determine the max and min of the objective functions
./min.C
```