OPTIMAL SCHEDULING OF BATCH CHEMICAL PROCESSES

by

Emilia Kondili

A thesis submitted for the Degree of

Doctor of Philosophy of the

University of London

Department of Chemical Engineering and Chemical Technology

Imperial College of Science and Technology

London

November 1987

To

my parents

and my brother

ABSTRACT

In batch chemical plants different products may be manufactured simultaneously; thus, they have to share the available resources. In order to optimize the operation of the plant, a scheduling problem must be solved.

The objective of this work is to develop a versatile technique for the optimal solution of the scheduling problem which arises in a general batch processing system.

A flexible network type representation is established which is sufficiently general to accommodate various special features of the batch chemical processing system. The mathematical formulation of the scheduling problem as a mixed integer linear programming model is generated. The flexibility of the proposed representation and mathematical formulation in dealing with a wide variety of characteristics of the problem is demonstrated.

A branch and bound type algorithm is developed to obtain the exact solution of the scheduling problem. Improvements of the efficiency of the algorithm by exploiting physical properties of the system are also described. Computational experience with the algorithm and comparison of its various versions are presented and the effects of various problem specifications are investigated.

Finally, the advantages of the proposed technique are discussed and the aspects that require further research are pointed out.

ACKNOWLEDGEMENTS

I would like to express my gratitude to Professor R.W.H. Sargent for his guidance, advice and encouragement during the course of this work and the preparation of this thesis. I am also deeply indebted to Dr. C.C. Pantelides for his invaluable assistance, patience and tireless contribution throughout all the stages of this study.

I am most grateful to Bodossaki Foundation (Athens, Greece) for their financial support for most of this study and to Onassis Foundation (Athens, Greece) for their contribution.

Many thanks are due to the staff of Chemical Engineering Department and Dr. Fred Marquis for their help.

Finally, I would like to express my most sincere thanks to all my friends in Montpelier Hall, and especially to Koula, Kevin, Ingrid and Hatim, for their understanding and consideration and for making my stay in London a very nice one.

CONTENTS

		Page
CHAPTER ONE	INTRODUCTION AND BASIC CONCEPTS OF SCHEDULING THEORY	
1.1	Introduction	1
1.2	Thesis Outline	3
1.3	Basic Concepts of Scheduling Theory	5
CHAPTER TWO	REVIEW OF LITERATURE	
2.1	Structure of Problems Arising in Batch Chemical Processes.	9
2.2	Review of Previous Work	11
2.2.1	Design and Scheduling of Multiproduct Plants	11
2.2.2	Design and Scheduling of Multipurpose Plants	14
2.3	Discussion	17
CHAPTER THREE	REPRESENTATION AND MATHEMATICAL FORMULATION OF THE SCHEDULING PROBLEM	
3.1	Description of the General Batch Chemical Processing System	18
3.2	The Concept of the State Task Network	20
3.2.1	Parameters associated with the State Task Network	23
3.2.2	Example 1. Construction of the State Task Network	24
3.3	Problem Definition	28
3.4	Assumptions	29
3.5	Mathematical Formulation	29
3.5.1	Representation of Time	30
3.5.2	Selection of Variables	30
3.5.3	Formulation of the Constraints and the Objective Function	32
3.5.4	The Complete Mathematical Model and its Dimensionality	42
3.5.5	Example	44
3.6	Concluding Remarks	48
	Nomenclature for Chapter 3	51

Page

CHAPTER FOUR THE SOLUTION OF THE MATHEMATICAL PROBLEM

4.1 Mixed Integer Linear Programming Systems 53

4.2 Branch and Bound Methods 55

4.2.1 The Concept of Branch and Bound 55

4.2.2 Computational Aspects of Branch and Bound Technique 59

4.3 The solution Strategy 61

4.3.1 Initial Solution Generating Algorithm 61

4.3.2 The structure of the Solution Algorithm 66

4.3.3 Branch and Bound Algorithm Version 1 71

4.3.4 Branch and Bound Algorithm Version 2 73

4.3.5 Branch and Bound Algorithm Version 3 84

4.4 Concluding Remarks 86

CHAPTER FIVE CASE STUDIES

5.1 Introduction 89

5.2 Batch System Problem 1 90

5.3 Batch System Problem 2 95

5.3.1 Batch System Problem 2a 101

5.4 Batch System Problem 3 105

5.5 Batch System Problem 4 109

5.6 Comparison of the Algorithm with a Commercial MILP Package 115

5.7 Discussion of the Results 116

CHAPTER SIX GENERAL CONCLUSIONS 120

REFERENCES 124

APPENDIX 1 Example for the Initial Solution Generating Algorithm 129

APPENDIX 2 Implementation Aspects 131

C H A P T E R O N E

INTRODUCTION AND BASIC CONCEPTS OF SCHEDULING THEORY

1.1 Introduction

Batch production has been the commonest procedure for the manufacture of chemicals of low volume and high commercial value. Because of the small quantities in which these products are manufactured, the assignment of a separate plant to each single product is economically unattractive. Therefore, batch chemical plants are usually flexible production facilities which can accommodate several different products in a variety of ways.

In batch chemical processes there is usually a large number of degrees of freedom available in deciding when and in what manner the products should be produced. On the other hand, the products being manufactured impose conflicting demands on the available resources of the plant. In addition, the transfer of material between the various stages of the process has to be coordinated in time as well as in quantity. Due to all the above reasons, batch processes are intrinsically more complex than continuous processes.

In a batch processing system a fixed sequence of tasks must be carried out to produce each of the products from its raw materials. The same equipment is used to process different tasks at different times, either in the same sequence or after some reconfiguration of the equipment. Among the decisions which must be made, taking into account a variety of constraints, are product amounts, the allocation of tasks to

equipment items and the size, type and number of these equipment items. Different types of problems are characterized by which variables are specified and which are free to be chosen to determine an optimal solution with respect to a performance criterion.

Design problems call for the selection of appropriate equipment items which will enable the product requirements to be met at minimum expenditure, which may include both equipment and operating cost.

In production planning and scheduling problems the plant specifications and the production requirements for a campaign are given. The amounts of the various products and the equipment modules in which the individual tasks should be carried out throughout the campaign are to be determined, so as to optimize some suitable economic or system performance criterion. The need for scheduling does not arise from the batch nature of the processing operations but from the fact that the available production time on each facility must be subdivided among the various processing tasks.

The solution of the scheduling problem will be critically affected by the structure of the processing network, the processing times for each product in each operation, the presence or absence of intermediate storage, the performance criterion used to rank schedules, as well as any due dates or production requirements which are assigned to individual products. A procedure aiming to give a realistic solution to the problem must be able to accommodate the above characteristics.

On the other hand, batch chemical plants are often subject to uncertainties present in the design phase, because of changes in product demands, and in the operation of the plant, due to changes in the availability and performance of equipment items. Uncertainties at the

design phase can be allowed for by initially building a plant with sufficient flexibility. Random disturbances in the operation of the plant can be dealt with to a limited extent by appropriate use of inventory.

Nevertheless, economic considerations call for increased efficiency in the design, planning and operation of batch chemical plants. From the above discussion it is concluded that the use of computer aided methods for the assessment of the performance of a batch plant is mandatory.

The scope of this study is to develop solution techniques for the optimal production planning and scheduling of batch chemical processes which will account for a large number of potential problem features. The problem is considered to be of deterministic nature without any stochastic variations in the parameters which define it. The aim of the study is fulfilled in three consecutive steps. Firstly, a suitable representation of the process is established. Then, the mathematical model of the problem is formulated. Finally, the algorithm which solves the mathematical problem is developed.

1.2 Thesis Outline

Scheduling problems are encountered in various practical situations and the early developments in the field of scheduling were motivated by problems arising in Operations Research and manufacturing. The rest of the first Chapter is devoted to a short description of the general scheduling problem and the methods which have been used to approach its solution.

The design and scheduling of batch chemical plants have only recently started to be studied systematically. In Chapter 2 the important contributions to this field of research are reviewed, following a more

detailed analysis of the structure of the problems encountered in batch processing systems. In Chapter 3 a new network type representation of batch processes is introduced and the mathematical model of the scheduling problem is formulated. The novel feature of the established representation is that, in addition to the processing tasks, the status of feeds, intermediate and final products is shown explicitly. Thus, potential complications of the system like mixed intermediate storage policy, recycling, splitting and mixing of material can easily be accomodated.

The mathematical formulation corresponds exactly to this representation and has resulted in a mixed integer linear programming (MILP) model which can accommodate a wide variety of objectives and constraints imposed on the system.

Chapter 4 is devoted to the description of the algorithms which have been developed for the solution of the mathematical problem. Branch and bound has been the most prominent method for solving MILP problems, especially when the structure of the system is exploited in order to curtail the search. The algorithm which has been developed employs the branch and bound technique to obtain the optimal solution of the problem. By utilizing a property of the system, the efficiency of the algorithm is improved substantially without compromising the optimality of the solution obtained.

Applications of the proposed technique and aspects of the computational performance of the algorithm are presented in Chapter 5. The final Chapter (sixth) reviews the thesis, highlights the parts which are believed to be of original contribution in the field and puts forward suggestions for further research.

1.3 Basic Concepts of Scheduling Theory

Scheduling is the allocation of resources over time to perform a collection of jobs. This problem arises in a variety of practical situations (eg. in Operations Research, manufacturing industry, batch chemical industry).

Scheduling theory is primarily concerned with the development of mathematical models and techniques that relate to the scheduling problem. Essentially the solution of such a problem is a feasible resolution of the constraints imposed on the system which optimizes the evaluation criterion. The feasibility constraints commonly met are limits on the capacity of available resources and restrictions on the manner in which assignments are made. On the other hand, decision making goals are mainly cost related measures or system performance criteria, like efficient utilization of resources, close conformance to prescribed deadlines and rapid response to demands.

A specific scheduling problem is described by four types of information:

1. the jobs to be processed and their precedence restrictions;
2. the number and type of resources (machines) that are available;
3. policies that restrict the manner in which assignments can be made; and
4. the criteria by which the schedule is evaluated.

A very frequent distinction made in scheduling environments is according to the flow of the work in the, so called, "shop".

In a flow-shop problem the flow of the work is unidirectional, ie. all the jobs to be processed follow the same path from one machine to another.

In the job-shop problem the flow of the work is not unidirectional and there is no common pattern of movement from one machine to another.

Scheduling problems are of a combinatorial nature with a discrete number of feasible solutions. However, the number of these solutions is so large even for simple cases, that their full enumeration is computationally prohibitive. The problem is often further complicated by the need to determine values of associated continuous variables.

A variety of techniques have been developed in order to solve scheduling problems. These solution techniques can be classified in two broad categories:

1. exact methods, which guarantee the optimum but can be inefficient for large size problems; and
2. approximate methods based on heuristics, which can obtain relatively good suboptimal solutions quickly but have little chance of obtaining the optimum.

All the exact solution techniques consist of a basic enumeration scheme, but a number of different formulations have been developed, like

i) enumeration of all feasible solutions;

ii) Dynamic Programming; and

iii) Integer Programming, being solved by cutting plane or branch and bound methods.

Approximate methods can be classified into

i) exact solutions of relaxed problems;

ii) ad hoc decision rules and algorithms; and

iii) incomplete search procedures.

The scheduling field has become a focal point for the development, application and evaluation of combinatorial procedures, mathematical

programming, network methods and heuristic solution approaches. The selection of an appropriate technique depends on the complexity of the problem, the nature of the model which describes it and the choice of the criterion used to evaluate the schedule.

The scheduling problem has associated with it a large literature and there are very good papers and books which review the various solution approaches (Baker, 1974; Conway, 1967; Graves, 1960). Most of this literature has its origin in non-chemical process applications and is not directly relevant to the present work; however, a review (Reklaitis, 1982) is available of the portions of the broad scheduling literature which are of most interest to Chemical Process Industry problems.

In general, the solution techniques developed for the flow-shop and job-shop scheduling problems are not directly applicable to batch chemical processes problems. Amongst the reasons is that some of the assumptions made are not realistic for chemical operations. For example, the assumption of strictly ordered operations which is valid in manufacturing industry is unrealistic for chemical processes, where splitting, mixing and recycling of material are very common. Other distinctive characteristics of batch chemical processing systems which must be taken into account are mixed intermediate storage policy, overlapping production cycles, multiple use of individual items in each cycle and allocation of parallel units at certain stages of the process. To the above it should be added that in scheduling problems of batch chemical processes the production quantities must be determined, in addition to the timing of the operations. This involves consideration of material balances and units capacity restrictions.

The above discussion leads to the conclusion that, although production planning and scheduling problems of batch chemical processes have many similarities with scheduling problems faced in other practical situations, they require special treatment which considers explicitly their inherent characteristics.

CHAPTER TWO

REVIEW OF LITERATURE

2.1 Structure of Problems Arising in Batch Chemical Processes.

In a batch chemical plant several products are manufactured from specified raw materials. The sequence in which the necessary operations (tasks) must be performed in proceeding from raw materials to final products is fixed for each product. The production is carried out in an available set of equipment modules. The equipment items which are suitable for performing each task are identified. The complexity of the batch processing problem mainly resides in the variety of alternative equipment configurations and in the manner in which the tasks of the various products are allocated to these configurations.

In design and scheduling problems of batch processing the system components which must be considered are:

1. the processing scheme, ie. the tasks, their sequence and processing times;
2. the equipment items which are available or, in design problems, must be constructed;
3. the availability of intermediate storage; and
4. the demand patterns for different products.

These components and their interrelations can be specified in many different ways and simplifying assumptions are usually made in order to reduce the degrees of freedom of the system.

The development of computer-aided methods for batch plants has lagged substantially behind the corresponding development for continuous plants. The main reasons are the complexity of batch processing and the fact that continuously operating plants have dominated the production of chemicals. However, in recent years renewed interest in computer aided design and operation of batch processes has been shown. Increased competition in batch chemicals production calls for improved design and more efficient use of existing batch facilities.

Before proceeding to review the most important of the existing work in the field, it is useful to mention the significance of some terms commonly used in batch processing.

In most of the studies dealing with batch chemical processes the production is assumed to be carried out in a number of independent batches, ie. quantities of material which undergo the whole sequence of tasks from raw materials to final products. The actual quantity of product that can be manufactured in one batch in the available equipment is defined as the batch size.

A batch chemical plant is usually classified according to the route which each product follows through the various processing units. In the multiproduct plant all the products follow the same path through the plant, they are produced one at a time and production runs or campaigns are carried out for each product in turn. Multiproduct plants are usually designed for a fixed set of products and independently of scheduling considerations.

In the multipurpose plant there is no common pattern of movement of the products through the plant. Even successive batches of the same product may take different routes, depending on the equipment

availability. Two or more products are produced simultaneously in the plant, and the set of products is not normally fixed. Multipurpose plants result in more efficient equipment utilization but are more complex to design, schedule and operate.

2.2 Review of Previous Work

The review which follows attempts to give an idea of the progress which has been made towards the solution of the design and the production planning problems arising in batch processing systems. In this review we will refer separately to the publications dealing with multiproduct and multipurpose plants. However, very good review papers exist, where the reported work is classified into categories according to the suggested solution technique and the various problem specifications (Rippin, 1983, 1985).

2.2.1 Design and Scheduling of Multiproduct Plants

The design of multiproduct plants deals with the determination of the number and size of the necessary equipment items and it is done independently of scheduling considerations. One of the first reported works in the design of batch chemical plants is that of Loonkar and Robinson (1970) which considers the problem of calculating the sizes of equipment items which minimize the capital investment of a single product batch plant. The same authors (1972) extend the analysis to multiproduct plants.

Sparrow, Forder and Rippin (1975) have developed a technique which can handle discrete equipment sizes and parallel equipment items. They employ two methods: a heuristic procedure using a weighted sum of product

requirements which allows a feasible solution to be obtained with little calculation, and a branch and bound method to obtain the optimal solution with capital cost as the objective function. The heuristic procedure provides a good initial solution for the branch and bound search. The development of a computer package (MULTI-BATCH) which incorporates the above ideas is reported by the same authors (1974).

Grossmann and Sargent (1979) formulated the design problem as a mixed integer nonlinear programme (MINLP) for which either the exact solution or a very good suboptimal solution is obtained, so that it is possible to circumvent a tedious branch and bound search. In their formulation they allow the processing times to be a function of the batch size.

Knopf, Okos and Reklaitis (1982) have extended the minimum cost batch plant design problem so that the effects of semicontinuous units can be taken into account.

The optimal retrofit design of multiproduct plants has recently been considered by Vaselenak, Grossmann and Westerberg (1987). The optimal addition of equipment to an existing plant is determined in view of changes of the product demands. The problem has been formulated as an MINLP, for which the global optimum is guaranteed.

The scheduling problem of multiproduct plants consists in determining the sequence in which the products are to be manufactured. Since all the products follow the same route in the plant, there are obvious similarities to the flowshop problem. However, in flowshop studies in the mechanical industry it is usually assumed that there is unlimited intermediate storage available between the operations. Thus, in a number of studies of the multiproduct plant scheduling problem based on the structure of the flowshop system, the necessary modifications have been

made in order to represent finite or no intermediate storage conditions.

Suhami and Mah (1981) have modelled the scheduling of a multiproduct plant as a flowshop scheduling problem with no intermediate storage, with the minimization of the time length required to complete all the jobs (makespan) as objective. A branch and bound technique has been employed which makes use of lower bounds based on statistical estimates of makespan from the completion of fixed partial sequences of job assignments. A heuristic rule is used to obtain an initial solution which serves as an upper bound to the optimum.

Knopf (1985) has formulated the problem in the same way, but considering finite intermediate storage to be available within the production facility. A branch and bound procedure has been developed which uses a simulation model to evaluate sequence completion times.

Lazaro and Puigjaner (1985) propose a methology to solve the medium term planning problem of a multiproduct plant using alternate dynamic programming and integer programming with restrictions. The same authors (1987) compared four different methods for solving the integer nonlinear cost optimization problem for the production planning of a multiproduct plant.

Ku and Karimi (1986) have formulated the scheduling problem of a multistage serial batch processing system with finite intermediate storage as an MILP with the minimization of makespan as objective function. For solving large size problems they have developed approximate algorithms to obtain suboptimal solutions under the same objective criterion.

A heuristic algorithm based on idle times of units has been developed by Rajagopalan and Karimi (1987) for the short-term scheduling of multiproduct plants with a mix of storage policies, non-zero transfer times and sequence dependent set-up times.

2.2.2 Design and Scheduling of Multipurpose Plants

The optimal design of a multipurpose batch plant is a more complicated problem than the multiproduct plant design. Since two or more products may be produced at any time, the solution of the production planning and scheduling problem is a necessary prerequisite to the consideration of the design problem of a multipurpose plant.

Suhami and Mah (1982) have formulated the optimal design problem as an MINLP. Parallel production of products which have no equipment requirements in common is allowed in the same campaign, but only one route is permitted for each product. The optimal or near optimal equipment configuration is selected with a partly heuristic procedure. The MINLP which corresponds to this configuration is solved using a reduced gradient code.

Related to the above work, Imai and Nishida (1984) have proposed another method for determining the equipment configuration. The method is based on the solution procedure of the set partitioning problem of Operations Research, for which several efficient algorithms are available. However, analysis in the application to large systems has not been done.

The same design problem has recently been considered by Vaselenak, Grossmann and Westerberg (1987). In their approach, the combinational problem of selecting the product grouping for the optimal schedule is

circumvented by a superstructure representation. The resulting formulation is similar to the problem of optimal design of a multiproduct plant and the solution of a single MINLP is required.

Another work related to the optimal design problem is that of Klossner and Rippin (1984). In their approach, a choice can be made between different production strategies, which involve consideration of the production of more than one product in one plant simultaneously, or distributing the production of the total set of products over several plants. The mathematical models lead to optimization with non-linear objective function and linear or non-linear constraints. The set partitioning problems arising in allocation of products to subgroups for parallel production in the same plant or independent plants are solved rigorously by branch and bound procedures.

In multipurpose plants more than one manufacturing route through the plant is available for each product. Thus, the scheduling procedure will have to select the route of each batch of material, as well as the sequence of batches.

Mauderli and Rippin (1979, 1980) have developed an algorithm which solves the production planning and scheduling problem of multipurpose plants with no intermediate storage in a number of consecutive steps. The procedure starts with an evolutionary enumeration of alternative production strategies which are subsequently screened by identifying and rejecting the inefficient strategies. Finally, an optimal production plan is obtained by solving an optimization problem, in which production time is allocated to the various campaigns. A computer package (BATCHMAN) has been developed to implement the procedure.

Rich and Prokopakis (1986) have formulated the sequencing and scheduling problem arising in multipurpose plants as a mixed integer linear programming model. In their formulation they assume that there is sufficient storage capacity in all the stages of the process and they consider fixed batch sizes. The mathematical programming problem is solved using commercial software.

Janicke (1984) has proposed the use of an exact partial algorithm which examines whether or not a batch with a given starting time can be scheduled. The algorithm can be incorporated into program systems for medium and long term production planning.

The same author (1987) has considered another subproblem of scheduling of multipurpose batch plants: the optimal assignment of products to equipment items, neglecting the order and sequencing of processing steps. This problem arises if differences exist in the suitability of equipment items for processing different tasks, and the optimal compromise between capacity and suitability is to be determined. The problem has been formulated mathematically as a capacitated transhipment model and standard mathematical software has been used.

The short term production scheduling of an existing multipurpose plant has been considered by Egli and Rippin (1987). The time allocation of batches is determined to meet delivery dates, taking into account a variety of local constraints, while incurring minimum cost arising from storage charges and product changeovers.

Intermediate storage plays a vital role in batch processes and it is used to increase plant availability, dampen the effects of process fluctuations, reduce process cycle times and provide flexibility in sequencing and scheduling. Takamatsu et al. (1982) have considered the problem of the optimal design and operation of a single product batch

process consisting of many batch stages and intermediate storage tanks. They formulated a deterministic model for the design problem and, by exploiting its characteristics, they calculated the number and size of equipment items and the volume of intermediate storage tanks which minimize the capital investment. Karimi and Reklaitis (1985) have derived analytical expressions for the limiting storage volume required to decouple the processing stages in a system composed of a set of parallel units followed by intermediate storage followed by another set of parallel units.

2.3 Discussion

From the preceding review it is evident that the solution techniques developed for the scheduling problem of a batch processing system are quite diverse and specific to problem structure. In many cases heuristic rules have been employed at certain stages of the solution technique. The advantage of heuristics is that a solution is obtained quickly, but there is often no reasonable way of establishing the quality of such solutions. Furthermore, the heuristic rules are problem dependent and they make the solution technique inapplicable even in slightly modified situations.

Another noteworthy point is that the problems treated involve a number of simplifying assumptions for the structure of the physical system. Although this seems to be necessary in order to reduce the very large number of potential alternative solutions, it reduces the generality of the proposed solution.

In conclusion, the production planning and scheduling problem of batch chemical processes is an open area for further study. In this context a new strategy has been developed which provides a systematic approach for solving a significantly more general form of the problem that previously possible.

C H A P T E R T H R E E

REPRESENTATION AND MATHEMATICAL FORMULATION OF THE SCHEDULING PROBLEM

3.1 Description of the General Batch Chemical Processing System

In the general batch operating system considered in this work, a set of products is to be produced from a set of feedstocks, according to a recipe, i.e. a prespecified sequence of elementary operations (tasks). The production lines may be cross-linked; in other words the products may share common intermediates. Splitting, mixing or recycling of material may occur at various stages of the process.

As a result of the processing scheme, feeds, intermediates and products all have predefined states and the processing tasks transform material from one set of states to another. In general, a task may have several inputs from given states in specified proportions. It may also produce several outputs at given states, again in specified proportions and at fixed times after the start of processing; these times may be different for each of the outputs.

The processing tasks are carried out in a set of available equipment items. Each item is capable of performing one or more task. Parallel items may be allocated to perform the same task in order to remove bottlenecks. The processing tasks have to share not only the equipment items, but also various other resources, like manpower and utilities, e.g. steam, cooling water and electricity.

The use of the same unit to produce different products may require that some amount of cleaning or other preparation be carried out between different processing tasks to ensure that sufficiently low impurity levels

are maintained. In many practical situations the extent of this cleaning depends on both the task most recently processed and the task about to be processed. In such situations it is not valid to absorb the cleaning time in the processing time of the tasks and the sequence-dependent cleaning times must be taken into account explicitly, when the time interval during which a task occupies a unit is calculated.

The intermediate products are either kept stored in storage tanks until a downstream unit becomes available or, if they are unstable or no storage is available, they proceed to the next task as soon as they are formed. Therefore, some subsections of the plant operate with finite intermediate storage (FIS) and others with no intermedate storage (NIS) available. Usually there is practically unlimited storage available for the feeds and the final products.

Various costs are incurred in batch processing systems, like the cost of new equipment in the design problem, cost of feedstocks, inventory and operating costs. An objective criterion under which problems of batch systems are commonly solved is the profit maximization, taking into account all the above factors. Another measure of the performance of a production strategy is its total output rate which is associated with the efficiency of equipment utilization.

The strategy developed for the solution of the scheduling problem of the batch system described above consists of three stages:

1. establishment of a suitable and flexible representation of batch processes;
2. identification and mathematical formulation of the problem to be solved; and
3. development of an algorithm to solve the mathematical problem.

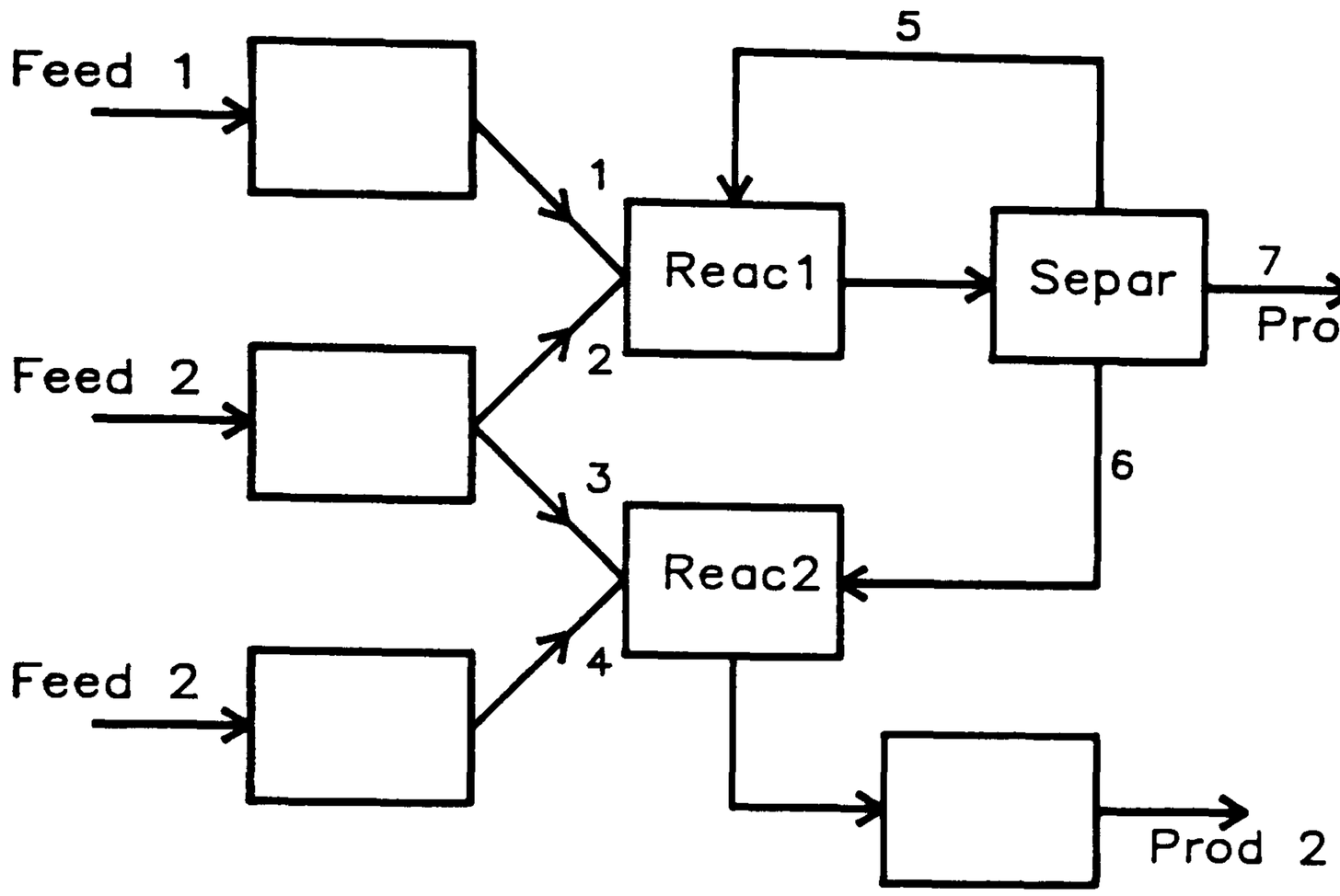
Each one of these aspects will be discussed in turn.

3.2 The Concept of the State Task Network

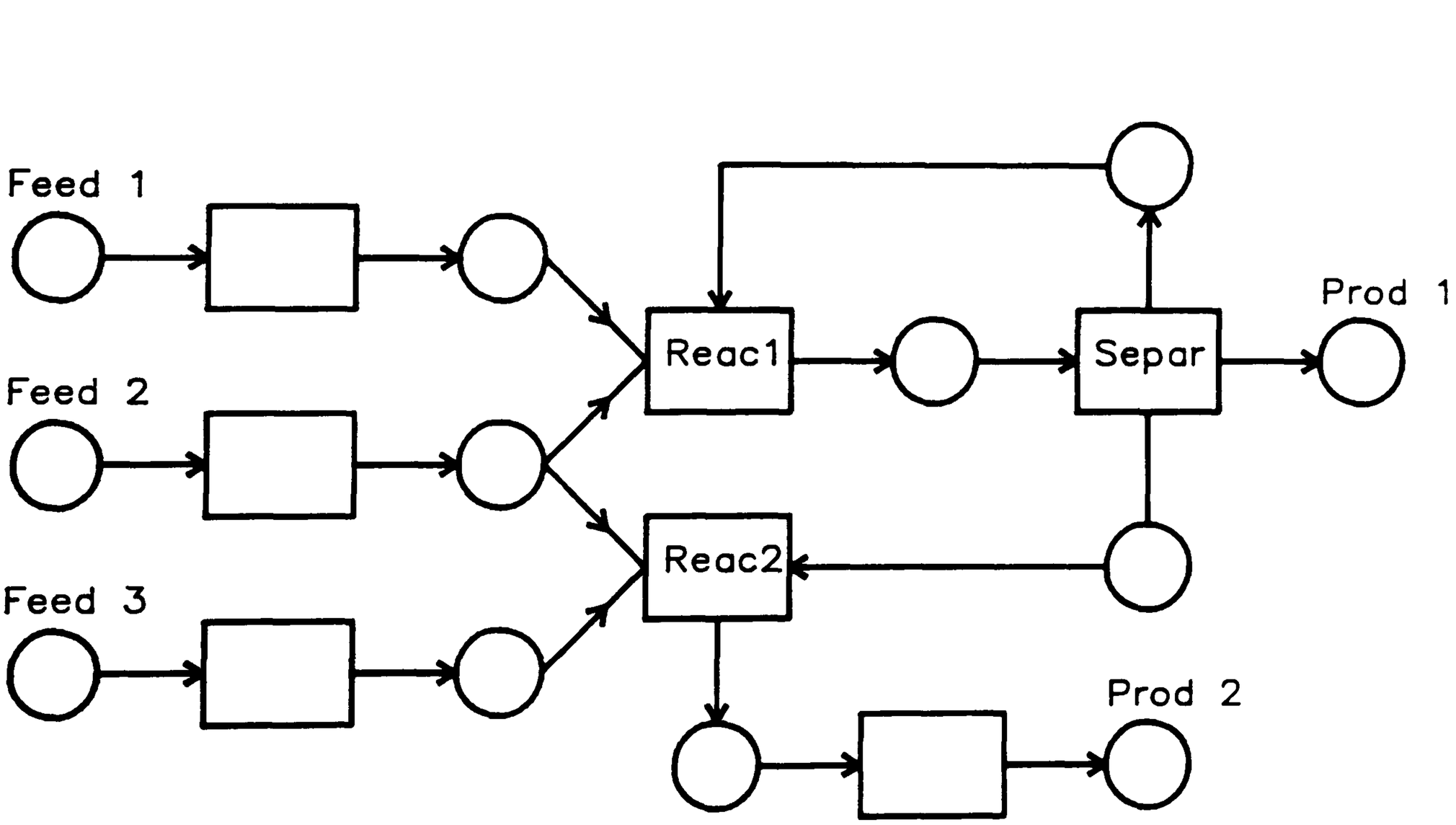
In order to represent the general batch operating system described in section 3.1, a suitable and flexible representation is needed. In the traditional, flowsheet-type representation of continuous processes only the processing tasks are shown explicitly. Although this representation is suitable for simple batch processes, it turns out to be inadequate for more complicated processes. This can be illustrated with a simple example.

Two products Prod1 and Prod2 are to be produced from feedstocks Feed1, Feed2 and Feed3 following the processing scheme shown in Figure 3.1a. In this process tasks Reac1 and Reac2 have more than one input stream. More specifically, task Reac1 has two input streams (1) and (2) from two preceding operations and input stream (5) which is a recycle from the separation task. The processing scheme of Figure 3.1a does not give any information on whether these three streams are identical or different streams. Furthermore, the separation task produces three output streams, (5), (6) and (7). Again no indication is given on whether the task essentially produces only one output which is split in three parts or three outputs of different quality. To the above it should be added that no information is provided on whether the material proceeds directly from one task to another or has the option to be kept stored between the processing stages.

The preceding discussion leads to the conclusion that the representation shown in Figure 3.1a does not describe the process sufficiently, since various ambiguities may arise. This problem can be overcome with a representation where, in addition to the processing tasks,



a. Conventional flowsheet



b. State Task Network

Figure 3.1 Representations of a batch processing system

the status of the various feeds, intermediates and final products is shown explicitly. This we shall call the State Task Network.

The processing scheme defines a Network which indicates the flow of material through the processing operations. The distinctive feature of the State Task Network is that it has two types of nodes; namely the State nodes, which represent the feeds, intermediate and final products and the Task nodes, which represent the processing operations. The states of the intermediate products can also be interpreted as intermediate storage stations, eventhough there may not be any storage available. The State nodes are denoted by a circle and the Task nodes by a box. The State Task Network (STN) which represents the process of Figure 3.1a is shown in Figure 3.1b.

In order to represent a batch processing scheme with a STN, a number of rules are followed:

1. A task has as many input (output) states as different input (output) streams.
2. Two or more streams entering the same state are necessarily of the same quality. If mixing of different streams is involved in the process, then it forms a separate task.

Although the process units are not shown explicitly on the STN, they appear in this representation implicitly, as will become evident in the following discussion. Generally, any batch chemical system, whatever complexities it may involve, can be described by a STN. The information which is usually available when dealing with batch processes (i.e. a sequence of tasks with fixed time requirements, the intermediate storage and the resources available) define the Network completely.

3.2.1 Parameters associated with the State Task Network

A State Task Network is described by its tasks, its states, and a set of available equipment items (or units). Each of these quantities has a number of parameters associated with it:

More specifically:

A Task i (i = 1, 2, ... NTasks) is defined by:

S_i : set of states which feed task i;

$\overline{S}_i$: set of states which task i produces as its outputs;

ρ_{is} : the proportion of input of task i from state $s \epsilon S_i$;

$$\left(\sum_{s \epsilon S_i} \rho_{is} = 1 \right)$$

$\overline{\rho}_{is}$: the proportion of output of task i in state $s \epsilon \overline{S}_i$;

$$\left(\sum_{s \epsilon \overline{S}_i} \overline{\rho}_{is} = 1 \right)$$

p_{is} : processing time for the output of task i to state $s \epsilon \overline{S}_i$;

$p_i = \max_{s \epsilon S_i} (p_{is})$ the completion time for task i; and

K_i : set of units capable of performing task i.

The last parameter correlates the process units to the STN.

A state s (s = 1, 2 ... NStates) represents material with given properties, which may be temporarily stored. Each state s is defined by:

T_s : set of tasks receiving material from state s;

$\overline{T}_s$: set of tasks producing material in state s;

C_s : maximum storage capacity for state s; and

C_s^o : the amount of material which is initially stored in state s

A process unit j (j = 1, 2 ... NUnits) may be capable of carrying out one or more tasks, depending on the requirements which the tasks place on the equipment. Each unit j is defined by:

I_j : set of tasks which can be performed by unit j; and

V_j : size of the unit.

The following example illustrates how the State Task Network corresponding to a batch operation and the parameters associated with it are derived, when all the necessary information is given.

3.2.2 Example 1. Construction of the State Task Network

The following batch chemical process is considered:

Products 1 and 2 are to be produced from feedstocks A, B and C.

Recipe:

- Task 1 (Heat) : Heat A for 1 hour;
- Task 2 (Reacl) : Mix 50% feed B and 50% feed C and let them react for 2 hours to form intermediate BC;
- Task 3 (Reac2) : Mix 40% hot A and 60% intermediate BC and let them react for 2 hours to form intermediate AB (60%) and product 1 (40%);
- Task 4 (Reac3) : Mix 20% feed C and 80% intermediate AB and let them react for 1 hour to form impure E;
- Task 5 (Separ) : Distill impure E to separate pure product 2 (90%, after 1 hour) and pure intermediate AB (10% after 2 hours). Recycle intermediate AB.

Available equipment:

- Unit 1 (Heater) : Capacity 100 Kg, suitable for task 1;
- Unit 2 (Reactor 1) : Capacity 50 Kg, suitable for tasks 2, 3, 4;
- Unit 3 (Reactor 2) : Capacity 80 Kg, suitable for tasks 2, 3, 4;
- Unit 4 (Still) : Capacity 200 Kg, suitable for task 5.

Available storage:

- For feeds A, B, C (States 1, 2, 3) : Unlimited
- For hot A (State 4) : 1000 Kg
- For intermediate AB (State 5) : 500 Kg
- For intermediate BC (State 6) : 0 Kgr
- For impure E (State 7) : 1000 Kg
- For products 1 and 2 (States 8, 9) : Unlimited

The above process is represented by the State Task Network shown in Figure 3.2.

The various parameters of the STN of the above process, as they are defined in Section 3.2.1 are:

* PROCESS TASKS

Task 1

S_1 = [feed A] , ρ_{11} = 1.0

$\overline{S}_1$ = [hot A] , $\overline{\rho}_{11}$ = 1.0

$p_{11} = p_1$ = 1 hr

K_1 = [Heater]

Task 2

S_2 = [feed B, feed C] , ρ_{22} = 0.5, ρ_{23} = 0.5

$\overline{S}_2$ = [IntBC] $\overline{\rho}_{26}$ = 1.0

$p_{26} = p_2$ = 2 hrs

K_2 = [Reactor1, Reactor2]

Task 3

S_3 = [hot A, IntBC] , ρ_{34} = 0.4, ρ_{36} = 0.6

$\overline{S}_3$ = [IntAB, Prod1] , $\overline{\rho}_{35}$ = 0.6, $\overline{\rho}_{38}$ = 0.4

$p_{35} = p_{38} = p_3$ = 2 hr

K_3 = [Reactor1, Reactor2]

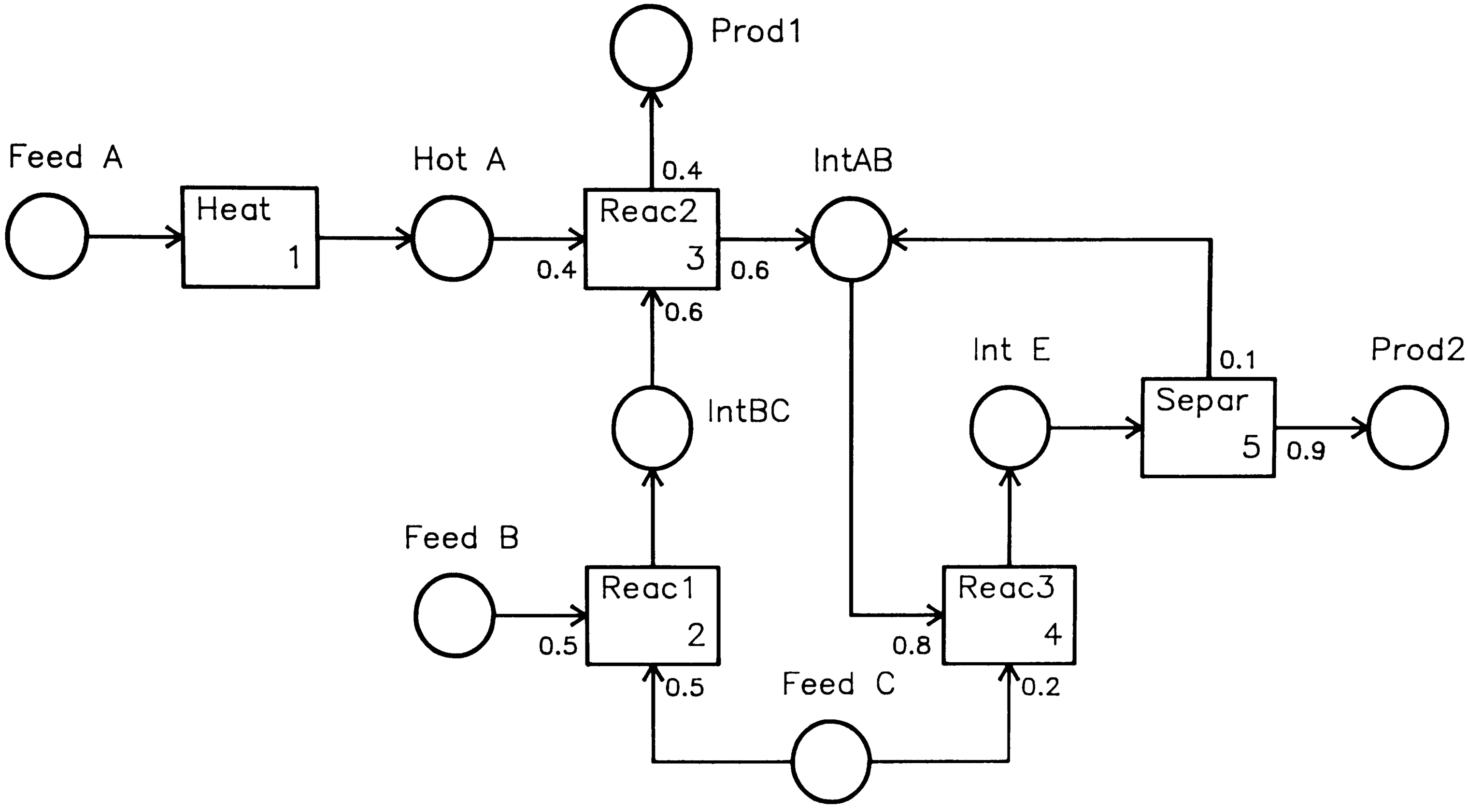


Figure 3.2 State Task Network for Example 1

Task 4 (Reac3)

S_4 = [feed C, IntAB] , $\rho_{43} = 0.2$, $\rho_{45} = 0.8$

$\overline{S}_4$ = [Impure E] , $\overline{\rho}_{47} = 1.0$

$p_{47} = p_4 = 1$ hr

K_4 = [Reactor1, Reactor2]

Task 5 (Separ)

S_5 = [Impure E] , $\rho_{57} = 1.0$

$\overline{S}_5$ = [IntAB, Prod2] , $\overline{\rho}_{55} = 0.1$, $\overline{\rho}_{59} = 0.9$

$p_{55} = 2$ hrs, $p_{59} = 1$ hr

$p_5 = \max(p_{55}, p_{59}) = 2$ hrs

K_5 = [Still]

* PROCESS STATES

State 1 (feed A)

T_1 = [Task 1]

$\overline{T}_1$ = empty

C_1 = unlimited

State 2 (feed B)

T_2 = [Task 2]

$\overline{T}_2$ = empty

C_2 = unlimited

State 3 (feed C)

T_3 = [Task 2, Task 4]

$\overline{T}_3$ = empty

C_2 = unlimited

State 4 (hot A)

T_4 = [Task 3]

$\overline{T}_4$ = [Task 1]

C_4 = 1000 kg

State 5 (IntAB)

T_5 = [Task 4]

$\overline{T}_5$ = [Task 3, Task 5]

C_5 = 500 kg

State 6 (IntBC)

T_6 = [Task 3]

$\overline{T}_6$ = [Task 2]

C_6 = 0

State 7 (Impure E)

T_7 = [Task 5]

$\overline{T}_7$ = [Task 4]

State 8 (Prod1)

T_8 = empty

$\overline{T}_8$ = [Task 3]

C_7 = 1000 kgr

C_8 = unlimited

State 9 (Prod 2)

T_9 = empty

$\overline{T}_9$ = [Task 5]

C_9 = unlimited

* PROCESS UNITS

Unit 1 (Heater)

I_1 = [Task 1]

V_1 = 100 kg

Unit 2 (Reactor1)

I_2 = [Task 2, Task 3, Task 4]

V_2 = 50 kg

Unit 3 (Reactor 2)

I_3 = [Task 2, Task 3, Task 4]

V_3 = 80 kg

Unit 4 (Still)

I_4 = [Task 5]

V_4 = 200 kg

3.3 Problem Definition

Having introduced the representation of a batch processing system with the State Task Network, the scheduling problem of a batch plant is defined as:

Given

- The State Task Network and all the information associated with it;
- a certain Time Horizon in which the scheduling problem is to be solved,

Determine

- The timing of the operation of each unit;
- the flow of material through the Network,

so that

an objective criterion is optimized.

3.4 Assumptions

In order to solve the problem as stated above, it is assumed that the batch plant operates under certain rules. More specifically:

1. No pre-emptive operation is allowed. Once processing of a task in a unit starts, the unit remains occupied until the task is completed.

2. Processing times of the task are fixed, independent of sizes and sequences.

3. Material is transferred instantaneously from states to tasks and from tasks to states.

4. All the numeric quantities involved are deterministic, without any stochastic variations.

5. There are not any unpredictable external or internal changes in the system during the time horizon considered (e.g. machine breakdown, unavailability of intermediate storage tanks etc).

3.5 Mathematical Formulation

Having established a suitable representation for the physical system, a mathematical model must be constructed which will enable us to analyze the problem in a systematic manner. In this respect three basic components are essential for formulating the mathematical model:

1. The parameter of time is inherent in any batch processing system. Thus, an appropriate timing mechanism, i.e. a representation of the time, must be found.

2. The dominant variables which are sufficient to describe the system must be identified.

3. The limitations of the system need to be translated into explicit constraints and the criterion under which the quality of the schedule will be evaluated into an explicit objective function.

3.5.1 Representation of Time

Traditionally, the cycle time, i.e. the largest processing time in the sequence of tasks for each product, has been used as the timing mechanism of batch operations. However, in the context of the State Task Network there are no independent production lines for each of the products to be manufactured; therefore another suitable time representation must be established. Thus, in our study we consider that:

1. Time is discrete. The time horizon is divided into a number of time periods of equal length.

2. All the events in the system take place on time interval boundaries only, not between them.

3. All processing times of the tasks (and cleaning times of the units, if there are any) are integer multiples of the elementary time quantum h. Actually the length of h is chosen in each case to be equal to the Greatest Common Divisor of all the processing times of the tasks (and cleaning times)

3.5.2 Selection of variables

In every mathematical model the variables must be chosen in such a way that they are sufficient to describe the system under consideration and, on the other hand, they keep the dimensionality of the problem within reasonable bounds.

The main aspects which characterize every batch processing system, irrespective of the objective to be optimized, are the allocation of units to tasks during the time horizon, the flow of material and the inventory levels through the process.

According to the above issues, the following variables are defined:

1. The decision variables dealing with the allocation of items to tasks. These are binary (zero/one) variables.

$$W_{ijt} = \begin{cases} 1 \text{ if unit } j \text{ starts processing task } i \in I_j \\ \text{at the beginning of time period } t \\ 0 \text{ otherwise} \end{cases}$$

2. The continuous variables L_{ijt}, dealing with the flow of material

   L_{ijt} : amount of material, which starts undergoing task i in unit $j \in K_i$ at the beginning of time period t

3. The continuous variables S_{st}, dealing with the amount of material being stored

   S_{st} : amount of material stored in state s at the beginning of time period t

The above definition of variables S_{st} allows the parameter C_s^o which represents the initial amount of material stored in state s to be expressed as S_{so}.

For convenience, the units of measurement of variables L_{ijt} and S_{st} are assumed to be the same as the dimension of the size of the units V_j.

While the quantities mentioned in Section 3.2.1 define the problem itself, variables W_{ijt}, L_{ijt} and S_{st} define the solution of the problem.

The most widely used graphical representation of scheduling relationships is the Gantt chart. It is essentially a graph of resource allocation over time. The resources (process units) are shown along the vertical axis and a time scale is shown along the horizontal axis, as in Figure 3.3. The line segments of the graph indicate the time intervals during which the corresponding unit is occupied processing a particular task. We will use this type of graph to illustrate the timing of the operations (i.e. the values of variables W_{ijt}) and, in an extended form of the graph, the corresponding amounts of material being processed (i.e. the values of variables L_{ijt}).

3.5.3 Formulation of the Constraints and the Objective Function

I. Constraints

The constraints which must be satisfied in our problem include:

1. The resolution of conflicts when allocating equipment items to processing tasks.
2. Limitations on the available resources.
3. Material balances.

In the following discussion we will show how these constraints can be put in mathematical form:

1. Allocation Constraints

The manner in which the process units are allocated to the tasks is restricted by two rules:

1. At any time each unit is either idle or processing a single task, and

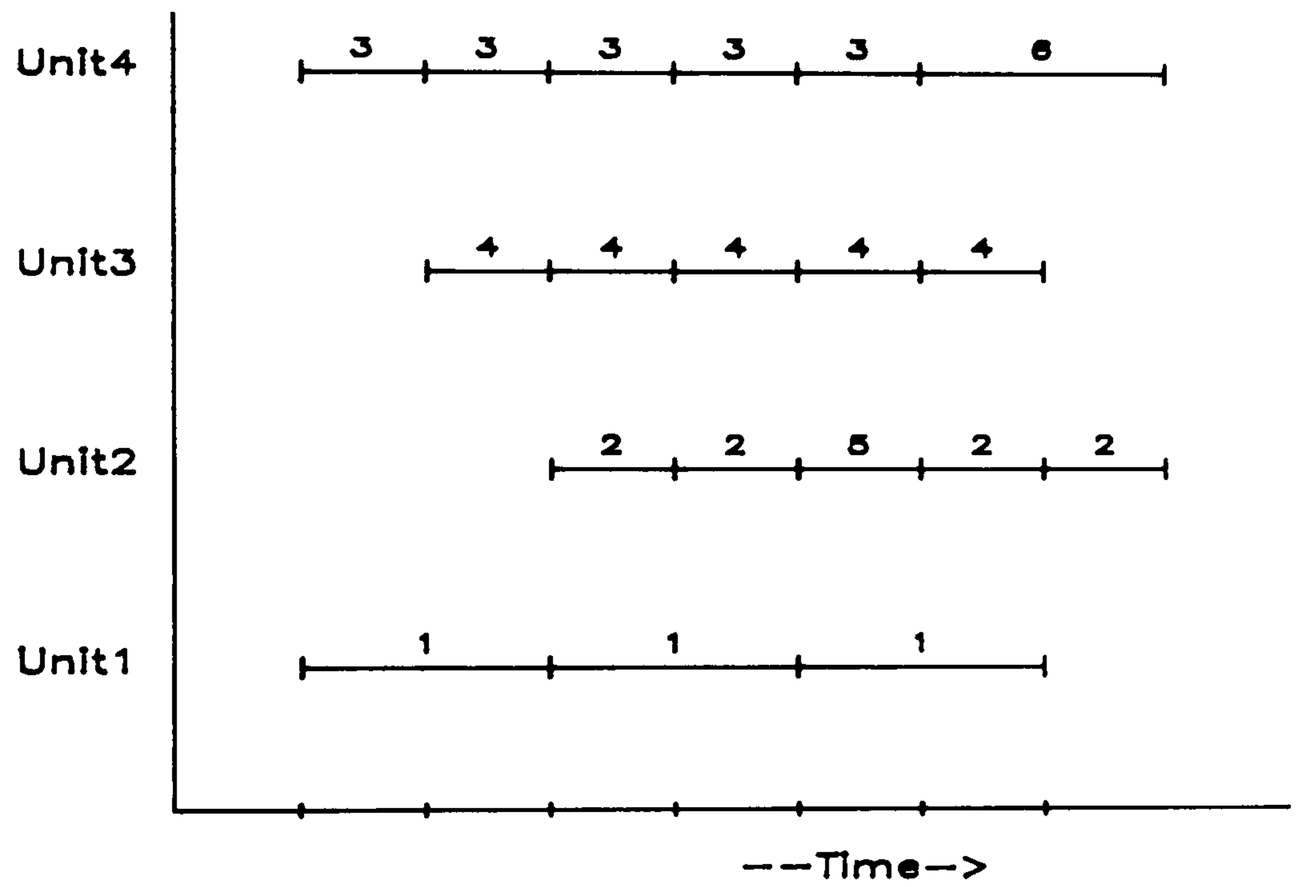


Figure 3.3 A Gantt chart

2. No tasks can be preempted once they have been released for production; i.e. once a unit starts processing a task i, it remains dedicated to that task for the next p_i time periods, until the task is completed. Both the above restrictions can be expressed mathematically as:

$$\text{if } W_{ijt} = 1 \text{ then } \sum_{t'=t}^{t+p_i-1} \sum_{i' \epsilon I_j} W_{i'jt'} = 1 \qquad (3.1a)$$

The summation over the time periods from t to $t+p_i-1$ ensures the continuity of task i (i.e. does not allow unit j to be preempted while task i is being processed). In addition, the summation over all tasks $i' \epsilon I_j$ excludes the starting of any other task in unit j during these time periods. The conditions expressed in constraint (3.1a) can be written in the form of a single linear constraint as:

$$\sum_{t'=t}^{t+p_i-1} \sum_{i' \epsilon I_j} W_{i'jt'} - 1 \leq M(1 - W_{ijt}) \quad \forall\, j,\ t,\ i \epsilon I_j; \qquad (3.1)$$

where M is a sufficiently large positive number, so that

$\sum_{t'=t}^{t+p_i-1} \sum_{i' \epsilon I_j} W_{i'jt'} - 1 \leq M$ becomes redundant with respect to any active constraint of the problem, e.g. $M = \max_j \left(\sum_{i' \epsilon I_j} \text{card}\,(I_j) \cdot p_{i'} \right)$. To show that (3.1) is a proper constraint, we consider all the possible alternative cases:

1. If $W_{ijt} = 0$, then (3.1) is a non-constraint
2. If $W_{ijt} = 1$ then the RHS of (3.1) is zero, forcing the LHS to zero,

   i.e. $\sum_{t'=t}^{t+p_i-1} \sum_{i' \epsilon I_j} W_{i'jt'} = 1.$

Since the summation term contains W_{ijt}, this means that no other $W_{i'jt'}$ is allowed to take a non-zero value.

Batch processes involve simultaneous operation of a set of units, each one in a different way at each stage of the process. The model being described is in accordance with the real system, where, as long as there are no conflicts, the process units operate independently of each other. Therefore, there is no restriction on the number of units assigned to the same task at the same time (parallel, in phase operation) or at different times (parallel, out of phase operation).

2. Capacity limitations

2.a Unit capacity constraints

The amount of material which starts undergoing task i in unit j at a time period t is limited by the size of this unit.

The mathematical expression of the above is:

$$0 \leq L_{ijt} \leq W_{ijt}.V_j \qquad \forall\ i,\ j\epsilon K_i,\ t \tag{3.2}$$

It may also sometimes be desirable to impose a lower bound on L_{ijt}, thus avoiding occupation of the unit with very small amounts of material, i.e.

$$\text{if} \quad W_{ijt} = 1, \quad \text{then} \quad f_j.V_j \leq L_{ijt} \leq W_{ijt}.V_j, \text{ or}$$

$$W_{ijt}.f_j.V_j \leq L_{ijt} \leq W_{ijt}.V_j\ ,\ 0 \leq f_j < 1 \tag{3.2a}$$

The total amount of material which starts undergoing task i at the beginning of time period t, L_{it}, is given by:

$$L_{it} = \sum_{j\epsilon K_i} L_{ijt}, \quad \forall\ i,\ t \tag{3.2b}$$

2.b Storage limitations

The amount of material stored in each state must not exceed at any time the maximum storage capacity of this state; i.e.

$$0 \leq S_{st} \leq C_s, \quad \forall\ s, t \tag{3.3}$$

3. Material balances

The amount of material stored in each state at a time period t equals the amount which was stored in the state at the previous time period plus what enters the state, i.e. the amount produced by tasks $i \varepsilon \bar{T}_s$ minus what exits the state, i.e. the amount of material, which starts undergoing tasks $i \varepsilon T_s$ at time period t.

In mathematical terms the above can be written as:

$$S_{st} = S_{s,t-1} + \sum_{i \varepsilon \bar{T}_s} \left(\bar{\rho}_{is} . L_{i,t-p_{is}}\right) - \sum_{i \varepsilon T_s} \left(\rho_{is} . L_{it}\right), \quad \forall\ s, t \tag{3.4}$$

In addition to the constraints (3.1) to (3.4), which are inherent in the system structure, other operational constraints may arise, possibly dictated by the particular case under consideration. There may be specified requirements of a particular product, limited feed or utilities availability, delays in raw material delivery etc. The corresponding constraints can easily be embedded in the model.

Although in our study we will take into account only constraints (3.1) to (3.4), we can illustrate how other operational constraints can be accommodated in the model with some examples.

Due dates

If a certain quantity Q_p of a product P must be supplied to a customer at a time $t_p < H$, then a constraint of the form

$$S_{P,t_P} \geq Q_P \qquad (3.5)$$

must be included in the model.

Cleaning times of the units

When a process unit is able to perform more than one task, situations may arise, where cleaning of the unit is necessary between processing of different tasks. The cleaning time of a unit is usually sequence dependent.

Suppose that the cleaning time is $a_{ii'j}$ if task i' follows task i in unit j. Then, the only change required is in constraints (3.1) which become:

$$\sum_{i' \in I_j} \sum_{t'=t}^{(t+p_i+a_{ii'j}-1)} W_{i'jt'} - 1 \leq M(1 - W_{ijt}), \quad \forall\, j,\ i \in I_j,\ t \qquad (3.1a)$$

where M is a sufficiently large positive number

Precedence constraints between products

In some cases the tasks are only allowed to be processed in a unit in a specified order (e.g. white products must be manufactured before coloured ones).

Suppose that a task $i \in I_j$ must precede all other tasks $i' \in I_j$, $i' \neq i$. Thus, if $W_{ijt} = 1$, all $W_{i'jt'}$, $i' \in I_j$, $t < t' \leq H$ must be equal to zero. Conditions like this can be handled by constraints (3.1a) by specifying a cleaning time $a_{ii'j}$ greater than H.

Limited availability of utilities

The processing tasks of a batch chemical plant have to share various utilities which may be of continuous nature (e.g. steam, cooling water

electricity) or discrete (e.g. manpower, auxiliary equipment). Limited availability of these utilities may cause conflicts, if they are required by the processing tasks simultaneously. In the following discussion we will illustrate how conditions of this type can be accommodated in the MILP model.

a. Continuous utilities requirement

In the general case the maximum available level of a utility may be time dependent (U_t). The (continuous) utility demand for task i carried out in unit j may have a fixed term and a variable term proportional to the amount of material being processed.

Thus, if $W_{ijt} = 1$, the contribution to the utility demand of the resulting operation in time step t+t', where $0 \leq t' \leq p_i-1$, is

$$d_{ijt'} + \tilde{d}_{ijt'} \cdot L_{ijt}$$

The total utility demand at each time period must not exceed the maximum available level of the utility. This condition can be expressed mathematically with the following constraint:

$$\sum_{i=1}^{NTasks} \sum_{j \in K_i} \sum_{t'=0}^{p_i-1} (d_{ijt'} \cdot W_{ij,t-t'} + \tilde{d}_{ijt'} \cdot L_{ij,t-t'}) \leq U_t, \quad \forall\, t \qquad (3.6)$$

b. Discrete utilities requirement

Normally a task i which starts taking place in unit j at time period t will require $N_{ijt'}$ discrete utilities, where t' counts from the start of the task, ie. $0 \leq t' \leq p_i-1$. In other words, if $W_{ijt}=1$, the contribution

to the discrete utility demand of the resulting operation in time step t+t', where $0 \leq t' \leq p_i - 1$, is $N_{ijt'}$.

The total discrete utility demand at each time period must not exceed the maximum available level of the utility. Mathematically, this can be expressed as:

$$\sum_{i=1}^{NTasks} \sum_{j \in K_i} \sum_{t'=0}^{p_i - 1} N_{ijt'} \cdot W_{ij,t-t'} \leq U^d_{t'} \quad \forall\, t \qquad (3.7)$$

II. The objective function

In order to complete the mathematical formulation we must define a criterion, which will assess the quality of a schedule. The model is capable of accommodating a variety of either economic or performance measurements. The criterion chosen for our formulation is the maximization of profit (which is equivalent to cost minimization). The principal types of costs than can be affected by scheduling decisions are the costs of feedstocks, inventory costs and operating and utilization costs of the units. The cost of a feedstock is proportional to the amount consumed.

Inventory costs are nontrivial and quite difficult to quantify; however, they are often in some way proportional to the physical level of inventory. Facility utilization is an important economic consequence of scheduling decisions. An efficient scheduling procedure will permit a given work load to be accomplished with a smaller aggregate demand on facilities.

In the context of the above discussion, our objective function can be expressed mathematically as:

max (Profit) or

max (Value of products - Cost of feedstocks - Inventory cost - Operating costs) (3.8)

Each of these factors is determined as follows.

a. Value of products = $\sum_{P}$ [(unit value of product) . (amount of product produced)]

The amount of a product P manufactured during the specified time horizon is given by the difference of the quantities stored in the corresponding product state at the end and at the beginning of the time horizon. Therefore,

$$\text{Value of products} = \sum_{P=1}^{NProd} c_P.(S_{P,H}-S_{P,0}),\ S_P\text{: product state} \qquad (3.8a)$$

Notice that the subscript 0 indicates the beginning of the time horizon and does not necessarily refer to time zero.

b. Cost of feedstocks = $\sum_{F}$ [(unit cost of feedstock) . (amount of feedstock consumed)]

The amount of feedstock F consumed during the time horizon is specified by the difference of the quantities stored in the corresponding feed state at the beginning and the end of the time horizon. Therefore,

$$\text{Cost of feedstocks} = \sum_{F=1}^{NFeeds} [c_F \cdot (S_{F,0} - S_{F,H})], \; S_F = \text{feed state} \tag{3.8b}$$

$$\text{c.} \quad \text{Inventory cost} = \sum_{I=1}^{NInter} c_I \cdot \sum_{t=1}^{H} S_{I,t} \; , \; S_I: \text{intermediate state} \tag{3.8c}$$

We have made the implicit assumption that there are no inventory costs associated with the feed and final product states. However, if these costs do exist, they can easily be incorporated in term (3.8c).

d. Operating costs.

The operating costs of the plant depend on the use of a particular process unit for a particular task and the amount of material being processed in each unit.

Thus, the total operating cost can be expressed as:

$$\text{Operating cost} = \sum_{t=1}^{H} \sum_{j=1}^{NUnits} \sum_{i \in I_j} (c_{ij} W_{ijt} + b_{ij} \cdot L_{ijt}) \tag{3.8d}$$

Other operating costs incurred in the plant and affected by scheduling decisions (e.g. cost of utilities) are usually small compared to the costs mentioned above. However, if constraints (3.6) and (3.7) are taken into account, a utility cost term should be introduced in the objective function.

In summary, the objective function to be optimized can be written:

$$\text{Max} \left[\sum_{P=1}^{NProd} c_P \cdot (S_{P,H} - S_{P,0}) - \sum_{F=1}^{NFeeds} [c_F \cdot (S_{F,0} - S_{F,H})] - \right.$$

$$\left. \sum_{I=1}^{NInter} (c_I . \sum_{t=1}^{H} S_{I,t}) - \sum_{j=1}^{NUnits} \sum_{i \epsilon I_j} (c_{ij} . \sum_{t=1}^{H} W_{ijt} + b_{ij} . \sum_{t=1}^{H} L_{ijt}) \right] \qquad (3.8)$$

As already mentioned, various other objective functions can be formulated and accommodated in the model.

Examples of alternative criteria are given in Chapter 5, in the context of the problems which are studied there.

3.5.4 The complete mathematical model and its dimensionality

The mathematical model which has been constructed and corresponds to the problem defined in section 3.3 is summarized below:

$$\text{Maximize } \left[\sum_{P=1}^{NProd} c_P . (S_{P,H} - S_{P,0}) - \sum_{F=1}^{NFeeds} c_F . (S_{F,0} - S_{F,H}) - \sum_{I=1}^{NInter} c_I . \right.$$

$$\left. \left(\sum_{t=1}^{H} S_{I,t} \right) - \sum_{j=1}^{NUnits} \sum_{i \epsilon I_j} \left(c_{ij} . \sum_{t=1}^{H} W_{ijt} + b_{ij} . \sum_{t=1}^{H} L_{ijt} \right) \right] \qquad (3.8)$$

subject to

$$\sum_{t'=t}^{t+p_i-1} \sum_{i' \epsilon I_j} W_{i'jt'} - 1 \leq M \left(1 - W_{ijt}\right) \quad \forall\, j, t, i \epsilon I_j \qquad (3.1)$$

$$0 \leq L_{ijt} \leq W_{ijt} . V_j \quad , \quad \forall\, i,\ t,\ j \epsilon K_i \qquad (3.2)$$

$$0 \leq S_{st} \leq C_s \quad \forall\, s,\ t \qquad (3.3)$$

$$S_{s,t} = S_{s,t-1} + \sum_{i \epsilon \overline{T}_s} \overline{\rho}_{is} L_{i,t-p_{is}} - \sum_{i \epsilon T_s} \rho_{is} L_{it} \quad , \forall\, s,\ t \qquad (3.4)$$

where $L_{it} = \sum_{j \epsilon K_i} L_{ijt}$

$$W_{ijt} \in (0, 1) \quad \forall\, j,\ i \in I_j,\ t$$

The above model is a mixed integer linear programming problem (MILP). The integer (binary) variables are the W_{ijt} and the continuous ones are the L_{ijt} and the S_{st}. The dimensions of the MILP, i.e. the number of variables and constraints, are given below.

I. Variables	Number
I.1 Binary variables	
W_{ijt}	$H \cdot \sum_j \text{card}\,(I_j)$
I.2 Continuous variables	
L_{ijt}	$H \cdot \sum_i \text{card}\,(K_i)$
S_{st}	$(H+1) \cdot \text{Nstates}$
Total	$H \cdot \left[\sum_j \text{card}\,(I_j) + \sum_i \text{card}\,(K_i) + \text{Nstates}\right] + \text{Nstates}$

II Constraints	Number
constraints of type (3.1)	$H \cdot \sum_j \text{card}\,(I_j)$
constraints of type (3.2)	$H \cdot \sum_i \text{card}\,(K_i)$
constraints of type (3.4)	$H \cdot \text{Nstates}$
Total	$H \cdot \left[\sum_j \text{card}\,(I_j) + \sum_i \text{card}\,(K_i) + \text{Nstates}\right]$

Actually, the term $\sum_j \text{card}\,(I_j)$ is equal to $\sum_i \text{card}\,(K_i)$, thus the number of constraints of type (3.1) is equal to those of type (3.2). The term H+1 in the number of S_{st} variables appears because we consider as variables the initial amounts of material stored in the states of the process. In fact these variables will be fixed.

In the evaluation of the constraints number, the constraints of type 3.3 are not taken into account, since they are simple bounds on the

variables and they do not complicate the problem any further. In any mixed integer linear programming problem the major factors of complexity are the gap and the number of the integer variables. In the above model the number of integer variables is determined by the number of processing tasks each of the available units is capable of carrying out, and the time horizon of interest.

The following example illustrates how the mathematical model of a batch process is constructed.

3.5.5 Example

Consider that the schedule of the process described in the example of section 3.2.2 over a horizon of 10 hours is to be determined, so that the profit is maximized. According to the above discussion, the mathematical model of the process is:

I. Objective function

$$\text{Maximize } \Big[\sum_{P=1,2} c_P \cdot (S_{P,H} - S_{P,0}) - \sum_{F=1,2,3} c_F \cdot (S_{F,0} - S_{F,10}) -$$

$$\sum_{I=4,5,6,7} c_I \cdot \sum_{t=1}^{10} S_{I,t} - c_{11} \cdot \sum_{t=1}^{10} W_{11t} - b_{11} \cdot \sum_{t=1}^{10} L_{11t} - \sum_{j=2,3} \sum_{i=2,3,4} \Big[c_{ij} \cdot \sum_{t=1}^{10} W_{ijt} + b_{ij} \cdot \sum_{t=1}^{10} L_{ijt} \Big] - c_{54} \cdot \sum_{t=1}^{10} W_{54t} - b_{54} \cdot \sum_{t=1}^{10} L_{54t} \Big] \text{, where } S_{P_1} = S_8,$$

$$S_{P_2} = S_9$$

II. Constraints

a. Allocation constraints

Unit 1 It can only perform task 1, with $p_1 = 1$ hr

time period 1 $W_{111} - 1 \leq M\,(1-W_{111})$

time period 2 $W_{112} - 1 \leq M\,(1-W_{112})$

time period 10 $W_{11,10} - 1 \leq M\ (1-W_{11,10})$

Unit 2 It can perform tasks 2, 3 and 4

Task 2, $p_2 = 2$ hrs

time period 1 $W_{221} + W_{222} + W_{321} + W_{322} + W_{421} + W_{422} - 1 \leq M\ (1-W_{221})$

time period 2 $W_{222} + W_{223} + W_{322} + W_{323} + W_{422} + W_{423} - 1 \leq M\ (1-W_{222})$

. . .

time period 10 $W_{22,10} + W_{32,10} + W_{42,10} - 1 \leq M\ (1-W_{2,2,10})$

All W_{ijt}, $t>H$ are assumed to be equal to zero.

Task 3 , $p_3 = 2$ hrs

time period 1 $W_{221} + W_{222} + W_{321} + W_{322} + W_{421} + W_{422} - 1 \leq M\ (1-W_{321})$

time period 2 $W_{222} + W_{223} + W_{322} + W_{323} + W_{422} + W_{423} - 1 \leq M\ (1-W_{322})$

time period 10 $W_{22,10} + W_{32,10} + W_{42,10} - 1 \leq M\ (1-W_{32,10})$

Task 4 , $p_4 = 1$ hr

time period 1 $W_{221} + W_{321} + W_{421} - 1 \leq M\ (1-W_{421})$

time period 2 $W_{222} + W_{322} + W_{422} - 1 \leq M\ (1-W_{422})$

. . .

time period 10 $W_{22,10} + W_{32,10} + W_{42,10} - 1 \leq M\ (1-W_{42,10})$

Unit 3 As for Unit 2, since units 2 and 3 can perform the same set of tasks.

Unit 4 It can perform task 5, with $p_5 = 2$ hrs

time period 1 $W_{541} + W_{542} - 1 \leq M\ (1-W_{541})$

time period 2 $W_{542} + W_{543} - 1 \leq M\ (1-W_{542})$

. . .

time period 10 $W_{54,10} - 1 \leq M\ (1-W_{54,10})$

b. Capacity Constraints

b1. Units capacity

Task 1 can be carried out only by unit 1

$0 \leq L_{11t} \leq W_{11t} V_1$

Task 2 can be carried out by units 2 and 3

$0 \leq L_{22t} \leq W_{22t} V_2$, $0 \leq L_{23t} \leq W_{23t} V_3$

Task 3 can be carried out by units 2 and 3

$0 \leq L_{32t} \leq W_{32t} V_2$, $0 \leq L_{33t} \leq W_{33t} V_3$

Task 4 can be carried out by units 2 and 3

$0 \leq L_{42t} \leq W_{42t} V_2$, $0 \leq L_{43t} \leq W_{43t} V_3$

Task 5 can be performed only by unit 4

$0 \leq L_{54t} \leq W_{54t} V_4$

$\forall\ t\ \varepsilon\ (1,10)$

b2. States capacity

For states 1, 2, 3, 8 and 9 the capacity is unlimited

State 4

$0 \leq S_{4t} \leq 1000$

State 5

$0 \leq S_{5t} \leq 500$

State 6

$S_{6t} = 0$

State 7

$0 \leq S_{7t} \leq 1000, \quad \forall\ t\ \varepsilon\ (1,10)$

c. Material balances

State 1 State 2

time period 1 $S_{11} = S_{10} - L_{11}$ $S_{21} = S_{20} - (L_{221} + L_{231})0.5$

time period 2 $S_{12} = S_{11} - L_{12}$ $S_{22} = S_{21} - (L_{222} + L_{232})0.5$

time period 10 $S_{1,10} = S_{19} - L_{1,10}$ $S_{2,10} = S_{29} - (L_{22,10} - L_{23,10})0.5$

State 3

time period 1 $S_{31} = S_{30} - 0.5\ (L_{221} + L_{231}) - 0.2\ L_{41}$

time period 2 $S_{32} = S_{31} - 0.5\ (L_{222} + L_{232}) - 0.2\ L_{42}$

time period 10 $S_{3,10} = S_{39} - 0.5\ (L_{22,10} + L_{23,10}) - 0.2\ L_{4,10}$

State 4

time period 1 $S_{41} = S_{40} - 0.4\ (L_{321} + L_{331})$

time period 2 $S_{42} = S_{41} + L_{111} - 0.4\ (L_{322} + L_{332})$

time period 3 $S_{43} = S_{42} + L_{112} - 0.4\ (L_{323} + L_{333})$

time period 10 $S_{4,10} = S_{49} + L_{119} - 0.4\ (L_{32,10} + L_{33,10})$

State 5

time period 1 $S_{51} = S_{50} - 0.8\ (L_{421} + L_{431})$

time period 2 $S_{52} = S_{51} - 0.8\ (L_{422} + L_{432})$

time period 3 $S_{53} = S_{52} + 0.6\ (L_{321} + L_{331}) + 0.1\ L_{541} - 0.8\ (L_{423} + L_{433})$

time period 4 $S_{54} = S_{53} + 0.6\ (L_{322} + L_{332}) + 0.1\ L_{542} - 0.8\ (L_{423} + L_{433})$

State 6

time period 1 $S_{61} = S_{60} - 0.6\ (L_{321} + L_{331})$

time period 2 $S_{62} = S_{61} - 0.6\ (L_{322} + L_{332})$

time period 3 $S_{63} = S_{62} + (L_{221} + L_{231}) - 0.6\ (L_{323} + L_{333})$

State 7

time period 1 $S_{71} = S_{70} - L_{541}$

time period 2 $S_{72} = S_{71} + (L_{421} + L_{431}) - L_{542}$

time period 3 $S_{73} = S_{72} + (L_{422} + L_{432}) - L_{543}$

State 8	State 9
$S_{81} = S_{80}$	$S_{91} = S_{90}$
$S_{82} = S_{81}$	$S_{92} = S_{91} + 0.9\ L_{541}$
$S_{83} = S_{82} + 0.4\ (L_{321}+L_{331})$	$S_{93} = S_{92} + 0.9\ L_{542}$
$S_{84} = S_{83} + 0.4\ (L_{322}+L_{332})$	$S_{94} = S_{93} + 0.9\ L_{543}$
$S_{8,10} = S_{89} + 0.4\ (L_{328}+L_{338})$	$S_{9,10} = S_{99} + 0.9\ L_{549}$

Dimensionality of the problem

(a)	Integer Variables	10 . (1+3+3+1)	= 80
	Continuous variables	10 . 8 + (10+1) . 9	= 179
	Total		259

(b) Constraints

$$10\ .\ (1+3+3+1) + 10\ .\ (1+2+2+2+1) + 10.9 = 250$$

3.6 Concluding Remarks

The representation of a batch process with the State Task Network and the mathematical formulation of the scheduling problem as an MILP enable us to deal in a natural manner with a number of constraints and/or objectives imposed on the plant.

Some of the characteristics of a batch operating system which can easily be accommodated are:

1. more than one product being manufactured simultaneously in the plant, not necessarily following independent production lines;

2. mixed intermediate storage policy within the same plant, depending on the capacity of storage stations which are available and/or on the nature of the intermediate products;

3. material being recycled, mixed or split between the various batch stages;

4. successive batches of material taking different routes in the plant, depending on the availability of items; the same equipment items may perform different tasks at different times;

5. more than one unit being allocated to the same task, operating in parallel in or out of phase, in order to relieve capacity or time bottlenecks;

6. processing tasks producing several outputs at different times after the start of processing (e.g. batch distillation);

7. individual tasks being performed depending on the availability of material in their input states, without the need to carry out the whole batch sequences;

8. sequence-dependent cleaning times of the units;

9. prespecified order in which a unit carries out different tasks;

10. various initial conditions being specified for the system;

11. a variety of optimality criteria based on either economic considerations (e.g. profit maximization, maximization of the production of a particular product for a certain time period), or performance criteria (e.g. maximization of equipment utilization);

12. a variety of operational constraints, such as contractual obligations (e.g. production of minimum amounts of some products, due dates).

From the above discussion it is concluded that the STN representation and the corresponding MILP model provide a general systematic framework for formulating a variety of different problems arising in batch processing systems with the same mathematical tool.

Another major feature of the MILP formulation for batch processes is that structural and parameter optimization are performed simultaneously.

This is a very important point since it avoids the need to iterate between the problems of finding the optimal configuration and evaluating the optimal flowrates and inventory levels, while satisfying the restrictions imposed on the plant.

NOMENCLATURE FOR CHAPTER 3

$a_{ii'j}$	cleaning time of unit j when task i' follows task i
b_{ij}	operating cost of unit j when performing task i
card(.)	cardinality operator (the number of elements in a set)
c_F	unit cost of feedstock stored in feed state F
c_I	inventory cost of intermediate state I per unit of stored material per time interval
c_{ij}	utilization cost of unit j when performing task i
c_P	unit value of product P
C_s	maximum storage capacity of state s
C_s^o	initial amount of material stored in state s
d_{ijt}	fixed term appearing in utility requirement
d_{ijt}	coefficient of variable term appearing in utility requirement
h	time quantum
H	time horizon
I_j	set of tasks which can be performed by unit j
K_i	set of items suitable for performing task i
L_{ijt}	amount of material which starts undergoing task i in unit j at the beginning of time period t
L_{it}	amount of material which starts undergoing task i at the beginning of time period t
NFeeeds	number of feedstocks
N_{ijt}	number of discrete utilities required by task i carried out in unit j at time period t
NInter	number of intermediate states
NProd	number of final products
NStates	number of states

NUnits — number of process units

p_i — processing time of task i

p_{is} — processing time for the output of task i to state $s \in \bar{S}_i$

S_i — set of input states of task i

$\bar{S}_i$ — set of output states of task i

S_{st} — amount of material stored in state s at the beginning of time period t

T_s — set of tasks requiring material from state s

$\bar{T}_s$ — set of tasks producing material in state s

U_t — maximum available level of a continuous utility at time t

U_t^d — maximum available level of a discrete utility at time t

W_{ijt} — decision variable indicating whether or not unit j starts undergoing task i at the beginning of time period t

Greek letters

ρ_{is} — proportion of input of task i from state $s \in S_i$

$\bar{\rho}_{is}$ — proportion of output of task i in state $s \in \bar{S}_i$

C H A P T E R F O U R

THE SOLUTION OF THE MATHEMATICAL PROBLEM

4.1 Mixed Integer Linear Programing Systems

Mixed Integer Linear Programming deals with problems in which a linear objective function is optimized, subject to both equality and inequality linear constraints. Two types of variables can be specified: continuous variables which can take any real value and discrete variables which can take integer values only (often only the values of 0 or 1). Its unique feature is precisely the capability of handling the latter type of variables which in application problems can be associated with discrete decisions.

In the case in which the integer variables can take only the values zero or one, the problem is of the form:

$$\text{maximize } z = \sum_{j \in N} c_j x_j \qquad (4.1)$$

subject to

$$\sum_{j \in N} a_{ij} . x_j \leq b_i, \; i \in M = \{1,2,\ldots,m\} \qquad (4.2)$$

$$x_j \in (0,1), \; j \in I \subseteq N = \{1,2,\ldots,n\} \qquad (4.3)$$

$$x_j \geq 0, \; j \in N - I$$

Several algorithms have been developed for the solution of this type of problem in the last two decades (Taha, 1975). However, none of the existing techniques is uniformly efficient from the computational standpoint, particularly as the size of the problem increases; their performance is highly problem dependent. Methods can generally be classified into two broad categories:

1. Cutting plane methods: These start with the continuous optimum, ie. the optimal solution of the linear programme obtained by relaxing the integrality constraints (4.3) to constraints of the form $0 \leq x_j \leq 1$. By systematically adding specially developed secondary constraints which essentially represent necessary conditions for integrality, the continuous solution space is gradually modified until its continuous optimum extreme point satisfies the integer conditions.

2. Search methods: these originate from the straightforward idea of enumerating all feasible integer points. Because the number of these points is, in general, extremely large, search methods are only practical if techniques can be developed which allow one to consider only a small portion of all the feasible solutions explicitly, while automatically discarding the remaining points as nonpromising. Clearly, the efficiency of the resulting search algorithm depends on the power of the techniques that are developed to discard the nonpromising solution points. Generally, search methods are more promising that the cutting plane techniques. The most prominent search method is the branch and bound which is described in more detail below.

4.2 Branch and Bound Methods

4.2.1 The concept of branch and bound

Branch and bound is an optimization technique where a selective search of possible solutions is carried out. The method involves enumerating a tree, each node of which represents a different combination of partial assignments of binary variables with fixed integer values. The enumeration of the nodes in the tree generates several feasible mixed integer solutions during the search. Assessing each node in the tree involves the solution of a linear programming subproblem that has some of the binary variables at fixed integer values, while the remaining binaries are treated as continuous variables. These linear programming subproblems are updated when solving for the different nodes, and their solution provides bounds for the solution of the original problem. By making use of these bounds, a selective search can be performed so that only a small subset of the nodes in the tree needs to be enumerated to find the global optimum solution.

The method consists of two fundamental procedures:

i. Branching is the process of partitioning the continuous solution space (obtained after relaxing the integrality constraints) into two or more continuous subproblems by fixing the values of one or more integer variables. The subproblems are themselves partitioned into further subdivisions, unless they yield a complete feasible solution (ie. one satisfying the integrality constraints). The purpose of partitioning is to eliminate parts of the continuous space, that are not feasible for the MILP problem.

The branching procedure replaces the original problem by a set of new problems, that are:

- mutually exclusive subproblems of the original;
- partially solved subproblems of the original; and
- smaller problems than the original.

ii. Bounding is the process of calculating a bound on the optimal solution of the objective function of a given subproblem. Assuming that the original problem is of the maximization type, the optimal objective value for each subproblem created by branching obviously sets an upper bound on the objective value attainable at any of its integer feasible points.

If a complete solution has been obtained at some stage and a subproblem encountered in the branching procedure has a performance measure worse (lower in the maximization case) than the complete solution, then the subproblem is not considered any further. In this case, the corresponding branch is said to be fathomed. By not branching any further from fathomed branches, the search process is curtailed although the entire set of feasible solutions is still enumerated implicitly. The search having been completed, the optimal solution is the best complete solution found.

A general algorithm for solving the MILP problem described in Section 4.1 is given below. The algorithm is initialized by obtaining an initial complete solution $\bar{z}_0$ by a heuristic technique. A solution is complete when all binary variables x_j, j = 1... Nint (Nint = number of integer variables) have integer zero/one values. This initial solution serves as lower bound to the optimal solution of the problem. Also in the initialization phase an upper bound to the solution is calculated by solving the LP obtained after relaxing all the integrality constraints.

The number of levels of the search tree is equal to the number of integer variables x_j (Nint). The branch and bound search is carried out by a recursive procedure Explore (j), j being the level of the tree where variable x_j is assigned zero/one values. The branching process, carried out in Step 1 of procedure Explore, comprises the generation of two subproblems, having variable x_j fixed to one and zero respectively. The upper bounds to the solution of these subproblems are calculated in Step 2.1, where the corresponding LPs are solved. In these LPs all variables x_i, $1 \leq i \leq j$ are fixed, while $0 \leq x_i \leq 1$, $j < i \leq$ Nint. In Step 2.2 the solution of the LP is compared against the best solution yet found. Eventually the procedure reaches Step **3**, where it checks whether the current level is the first one; if it is, the search has been completed, otherwise it backtracks to the previous level and checks whether there are any unexplored nodes.

The procedure always terminates, since all the solutions encountered at level Nint are complete. Earlier termination of the procedure may occur if a complete solution is encountered (Step 2.3) or if the branch is fathomed (Step 2.2).

The basic steps of the algorithm are given below:

<u>Algorithm G1</u>

1. Initialization phase

1.1 Construct an initial complete solution by a heuristic technique.

Set BestSolution = $\bar{z}_0$

1.2 Relax all integrality constraints. Solve the corresponding LP.

If infeasible then

Stop. So is the MILP

else

If solution complete then

Stop. This is the optimal solution of the MILP.

else

2. Explore (1)

3. Stop

Algorithm G1.1

Procedure Explore (j).

1. Create 2 subproblems by assigning values zero and one to a variable x_j, $0 \leq x_j \leq 1$, which has not been fixed yet.

2. For subproblems generated in Step 1 do

2.1 Solve corresponding LP. Let z(LP) be the value of the objective function obtained.

2.2 If z(LP) > BestSolution then

2.3 if solution complete then

Set BestSolution = z(LP)

else

if j < Nint then

Explore (j + 1).

3. Return

In many respects the above Algorithm G1 is a general prototype of a whole spectrum of branch and bound methods and it pays no regard to the

intrinsic properties of the problem being solved. Consequently, it tends to take longer than branch and bound algorithms specially designed for particular classes of problems.

4.2.2 Computational aspects of branch and bound technique

The total amount of computation required in a branch and bound method is related to the number of distinct bounding problems created and, hence, to the total number of nodes in the fully developed tree. Therefore, the most effective measure of the efficiency of the technique is the total number of Linear Programming problems to be solved. The amount of temporary storage required is related to the maximum number of leaves of the tree at any intermediate stage of its development. In the following discussion we describe how various strategies may be employed to reduce the amount of computation at the expense of additional storage and vice-versa.

Generally, the power of the technique mainly depends on the quality of the bounds obtained, since good bounds eliminate nodes high up in the tree, thus reducing the search substantially. The complete description of a branch and bound algorithm requires:

a. a specification of the rule that determines which of the currently active problems is to be chosen for branching; and,

b. the method for deriving new bounding problems.

As far as the first of these issues is concerned, the two most extensively used policies are:

a. branch from the newest active bounding problem, and

b. branch from the best bound.

a. **Branch from the newest active bounding problem**

The idea is always to branch from the latest problem created by branching (depth-first search). This strategy can be implemented with a 'pushdown' list. Each time branching is carried out, the new problems are placed on the top of the list and those below are 'pushed' down. The problem currently on the top of the list is examined. If it is active (ie. its bound is better than the best solution yet found), branching is carried out upon it; otherwise it is discarded and the next problem on the top of the list is examined. This policy has the distinct advantage that, for most problems, a minimum of computer storage is required. However, many more branching operations may have to be performed than under the policy which is described below.

b. **Branch from best bound**

This policy is to branch from that bounding problem which currently has the best bound (breadth-first search) and it has the advantage that in most cases the total amount of computation is minimized. However, the volume of intermediate data that must be stored may be quite large.

The conclusion from the above discussion is that there is no definite best way for selecting the branching variable or the specific sequence in which the subproblems must be scanned. Empirical rules exist, mainly developed through experimentation. These often enhance the process and are implemented in most of the commercial general purpose branch and bound codes. The latter perform well as long as the number of binary variables is not too large (typically a maximum of 200). When dealing with large-scale systems, the structure of the particular problem under consideration must be taken into account, in order to increase the efficiency of the branch and bound technique.

4.3 The Solution Strategy

The dimensionality of the MILP model describing our problem makes the cost of using existing general purpose MILP packages prohibitive (*). Therefore, a specially designed schedule generating algorithm has been developed, using the branch and bound technique to find the optimal solution of the problem. The algorithm is described in detail in Sections 4.3.2 to 4.3.5.

At any stage of the branch and bound method the best complete solution yet found is used in comparisons and allows branches of the search tree to be fathomed. In the initialization phase of the tree search a trial solution can be obtained by pursuing the tree directly to the bottom as rapidly as possible, or, at the very outset, by applying a heuristic procedure. In many cases the speed of finding an optimal solution can be increased if the search starts with a good trial solution, since many nodes may be eliminated in the early stages.

For the above reasons, in addition to the branch and bound algorithm, an efficient procedure has been developed to provide an initial solution to our problem. This is described in the following section.

4.3.1 Initial solution generating algorithm

The problem under consideration consists of two coupled subproblems:

i) the allocation of equipment items to processing tasks throughout the specified time horizon; and

* However, some experiments have been performed using one of them to test the validity of the model.

ii) the evaluation of the batch sizes (ie. the amounts of material undergoing each processing task) and the inventory levels with respect to a certain unit configuration. In the search of the optimal solution these subproblems are interrelated, since the optimal configuration can not be determined independently of the capacity restrictions and the way in which the material flows in the plant. However, this distinction proves useful for the purposes of this section.

If for a particular problem all the units are allocated to processing tasks over the time horizon of interest, so that no unit allocation constraints are violated, all binary decision variables will have fixed values W^{*}_{ijt}. Then the mathematical problem of Section 3.5.4 becomes a simple linear programme, the solution of which determines the optimal values of the continuous variables L_{ijt} and S_{st} for the assumed unit configuration.

More specifically, the only constraints of the system are those of type (3.4), since the units allocation constraints (3.1) no more exist and the units capacity constraints (3.2) are simple bounds on variables L_{ijt}. Therefore, the model will be of the form:

$$\text{maximize} \quad \sum_{P=1}^{NProd} c_P.(S_{P,H} - S_{P,0}) - \sum_{F=1}^{NFeeds} c_F.(S_{F,0} - S_{F,H}) -$$

$$\sum_{I=1}^{NInter} c_I.\sum_{t=1}^{H} S_{I,t} - C_{util} - \sum_{j=1}^{NUnits} \sum_{i\epsilon I_j} b_{ij} \sum_{t=1}^{H} L_{ijt}, \qquad (4.4)$$

$$\text{where} \quad C_{util} = \sum_{j=1}^{NUnits} \sum_{i\epsilon Ij} c_{ij} \sum_{t=1}^{H} W^{*}_{ijt}$$

subject to

$$0 \leq L_{ijt} \leq \bar{L} \qquad \forall\, i,j \in K_i, t \qquad (4.5)$$

where $\bar{L} = W^*_{ijt} \cdot V_j$, W^*_{ijt} fixed.

$$0 \leq S_{st} \leq C_s \quad , \forall\, s,t \qquad (4.6)$$

$$S_{st} = S_{s,t-1} + \sum_{i \in \bar{T}_s} \bar{\rho}_{is} \cdot L_{i,t-p_{is}} - \sum_{i \in T_s} \rho_{is} \cdot L_{it}, \quad \forall\, s,\ t \qquad (4.7)$$

From the preceding analysis it is concluded that if we determine a feasible solution to the units allocation problem for a given time horizon and then solve the above LP, we will obtain a value of the objective function which can be used as initial (trial) solution in our search. A procedure has been developed which allocates equipment items to tasks, ie. determines the timing of the operation of each unit. Although the procedure does not consider all the quantitative constraints of the system, it does take into account properties and restrictions qualitatively, so that only non-trivial feasible solutions are generated.

The principles under which the procedure is developed are:

1. The process is considered to be carried out as a number of passes of material through the State Task Network (batch sequences).

2. The number of batch sequences depends on the length of the Time Horizon. The last task of the last sequence must be completed at time ≤ Time Horizon.

3. Each batch sequence is scheduled independently of all the subsequent ones.

4. In each batch sequence:

   a. All the tasks take place in their prespecified order.

   b. Each task takes place exactly once and is only executed when there is material available in all its feed states.

   c. Each task is allocated only one equipment item.

   d. Storage feasibility is checked qualitatively (see below).

5. Whenever more than one unit can be allocated to the same task, the unit is chosen arbitrarily.

The basic structure of the algorithm is given below.

Algorithm A0

1. Set ibatch = 0

2. Set all units available at all times.

3. Repeat

   3.1 Assign Tasks To Units (Assignment-possible).

   If Assignment-possible then

   3.2 set ibatch = ibatch + 1

   3.3 update unit availability

   Until not Assignment-possible.

4. Solve the corresponding LP to evaluate the processed amounts of material, the inventories and the value of the objective function.

Conflicts in the use of equipment items are avoided by allowing them to be allocated to tasks in Step 3.1 only in the time intervals during which they are available. These intervals are identified in Step 3.3.

Storage feasibility is checked qualitatively in Step 3.1. Thus, if an intermediate state has no storage capacity available, the tasks producing output in this particular state are synchronized with the ones which require material from it.

In Step 3.1 an equipment item is found for each task for the current batch sequence. If no more batch sequences can be scheduled in the specified time horizon, the variable Assignment-possible becomes false. When the solution found is infeasible (ie. violates storage constraints), Step 3.1 (which is performed by a recursive procedure) retraces the solution path (backtracks) and chooses an alternative solution.

An example showing how Algorithm A0 proceeds is given in Appendix A.

The quality of the solution obtained may not be very good since the choice of units to be allocated to processing tasks is done arbitrarily, without any priority rules. In addition, although the algorithm provides the units with the option of remaining idle for a certain time period, while they could process a task, there are no quantitative criteria under which the units would choose this option. However, the algorithm is very efficient since the assignment step does not require any significant computational time and only one LP problem is being solved. Thus, it provides quickly a complete solution which can be of value at the early stages of the branch and bound search.

Examples of the application of the algorithm are given in Chapter 5.

The initial solution generating algorithm, as well as the branch and bound algorithm described later, have been implemented as a PASCAL program

linked to the MINOS large-scale linear programming code. The input information is supplied in the form of a specially designed language, which corresponds closely to the State Task Network and is described in Appendix B.

4.3.2 The structure of the solution algorithm

The basic concepts of branch and bound are applied in order to solve the scheduling problem of a batch chemical process, as defined in Section 3.5.4. The search tree is constructed in such a way that:

a. Each level of the tree represents a certain time period. Consequently, the total depth of the tree (ie. the number of levels) is equal to the time horizon of interest;

b. Each node of the tree represents a partial assignment of the process units to tasks. In other words, at a level T each subproblem generated has all the units allocated to tasks for all time periods $t \leq T$. Referring to the MILP model describing our problem, this means that all $W_{ijt}, t \leq T$ have been assigned fixed zero/one values.

Exhaustive search of the branching tree would be equivalent to complete enumeration of all unit configurations. An upper bound $\bar{z}$ on the value of the objective function can be calculated for each node at level T of the tree by solving the associated linear programming problem. This LP is obtained by fixing the W_{ijt} variables to their corresponding values for all $t \leq T$, while relaxing the integrality constraints on all other W_{ijt} (ie. $0 \leq W_{ijt'} \leq 1$ for $T + 1 \leq t' \leq H$). If a complete solution (ie. a solution with all binary variables having fixed zero/one values) is available with a value of objective function z^* and $\bar{z} > z^*$, then the

search proceeds to the next level. On the other hand, if $\bar{z} < z^*$, then no completion of the partial sequence σ^T could ever achieve a better performance and , therefore, the branch is not considered any further.

The algorithm consists of the initialization phase, where an initial solution to the problem is obtained (see Section 4.3.1) and a recursive procedure Search (T), where subproblems at a particular time level T are generated, evaluated and either discarded or branched further to the next level. The basic structure of the algorithm is given below.

Algorithm A.1

1. Construct a trial solution using Algorithm A0.

   Set BestSolution = $\bar{z}_{init}$

2. Search (1)

3. Stop

Algorithm A.1.1

Procedure Search (T)

1. Generate all feasible assignments at time T, for given assignments at time T - 1.

2. Obtain upper bound on solution of each assignment.

3. Determine order in which assignments are to be scanned.

```
4.  For each assignment generated do
    (4a)  if Upper Bound > BestSolution    then
          (4b) if solution complete    then
               (4c)    set BestSolution = Upper Bound
               (4d) else
               (4e)    if T < time Horizon    then
               (4f)       Search (T + 1)
5.  Return
```

Note that the procedure always terminates because all solutions obtained at time T = H are complete. However, it may terminate earlier either because a complete solution is found (Step 4b) or the branch is cut (Step 4a).

In Algorithm A.1.1 the subproblems are generated in Step 1 and their upper bounds are calculated in Step 2. The order in which the nodes are to be scanned is determined in Step 3 and the comparison of the subproblems against the best solution yet found is carried out in Step 4. In Step 5 the procedure retraces the solution path by returning to the node from which the current node has been created. The algorithm terminates when all the solutions have been enumerated.

Based on this structure, three versions of the algorithm have been developed, differing in the way the various subproblems generated in Step 1 are evaluated (Step 2) and scanned (Step 3). In Table 4.1 the main differences of the three versions of the algorithm are highlighted.

Table 4.1 Basic differences of the three versions of the algorithm

	Evaluation (Step2)	Ordering (Step 3)
Algorithm BB1	Solution of LP	Natural order.
Algorithm BB2	Solution of LP and dominance property utilization	Natural order.
Algorithm BB3	Solution of LP and dominance property utilization	In non increasing order of upper bounds.

The construction of the various subproblems of the search tree is performed by Step 1 of Algorithm A.1.1 and is essentially the same for all the versions of the algorithm. Given a set of assignments for time T - 1, this first step allocates all the units to process tasks at time T, considering all the potential alternatives in turn. Only feasible allocations are made; ie. if the units are available at time T, then they are allocated to tasks having material available in all their feed states. The option of each unit remaining idle at time T is also considered. Initially all the units are available at all times. With regard to the mathematical model, the construction of a subproblem implies that a number (equal to the number of process units) of binary variables is being fixed.

The generation of all the alternative feasible assignments at time T for a given assignment at time T - 1 is performed by a recursive procedure AssignUnit (junit, T). The procedure checks whether the unit is available at time T and, if it is, it attempts to find a task (itask) which can be

performed by the unit by scanning the list I_{junit}. If there is such a task which has material in all its feed states, then the unit is allocated to the task. This information is stored in an array AssignArray [Nunits, TimeHorizon], (all the elements of which are set to zero in the initialization phase), ie. AssignArray [junit, T] = itask. When the algorithm backtracks, the element AssignArray [junit, T] is set back to zero.

The availability of units is kept track of with a variable "unitavailtime" associated with each unit. Initially this variable is set equal to zero for all the process units. In addition a variable "stateavailtime", associated with each state, is used to indicate the soonest time that a state receives material. Initially it is set to infinity for all the states except the feed states, in which it is set equal to zero.

The procedure is called by Search (T) as AssignUnit (1, T) and has the following structure.

Algorithm A.2

```
GenerateAssignments
begin
   AssignUnit (1,T).
end
```

Algorithm A.2.1

Procedure AssignUnit (junit, T).

If junit ≤ NumberOfUnits then

1. if unit [junit].unitavailtime ≤ T then

for all tasks such that itask ε I_{junit} do

if state [istate].stateavailtime ≤ T for each istate ε S_{itask}

then

(1a) set AssignArray [junit,T] = itask

(1b) set unit[junit].unitavailtime = T + p_{itask}

(1c) for all states such that jstate ε $\bar{S}_{itask}$ do

set state [jstate].stateavailtime =

= min (stateavailtime, T + $p_{itask,jstate}$)

(1d) AssignUnit (junit + 1, T)

(1e) Set AssingArray [junit, T] = 0

(1f) Reset unit[junit].unitavailtime and

state [jstate].stateavailtime for all jstate ε $\bar{S}_{itask}$

(1g) Return

2. AssignUnit (junit + 1,T)

3. Return

Note that by calling procedure AssignUnit for junit + 1 in Step 2, junit remains idle at time T since the element AsssignArray[junit, T] has been set to zero in Step (1e).

4.3.3 Branch and bound algorithm Version 1

In Step 2 of Algorithm A.1.1 (which is essentially the bounding process) a linear programming problem is solved for each subproblem considered and provides the necessary upper bound. This LP is obtained by

fixing all the binary variables W_{ijt} for $t \leq T$ in the MILP model of Section 3.5.4 and relaxing the integrality requirements in all other W_{ijt}.

The nodes of the tree are scanned in the order in which they are generated. Thus, Step 3 of Algorithm A.1.1 is essentially omitted. Therefore, it is not strictly necessary to generate all subproblems at Step 1 before proceeding to the evaluation Steps 2 and 4.

In Step 4 the process attempts to fathom the branch it currently pursues by utilizing the best complete solution yet found. If fathoming is not possible, it may reach another complete solution or it may proceed to the next level. Eventually the algorithm reaches Step 5 and retraces the solution path upwards (backtracking) until it finds a level at which an unexplored node remains. Then Step 1 creates a new assignment and the algorithm proceeds again towards the last level. The advantage of the above depth-first search technique is that it does not have any severe storage requirements, since there is only one active subproblem at a time.

The first version of the algorithm is summarized below:

Algorithm BB1

1. Construct a trial solution using Algorithm A0
   Set BestSolution = z_{init}
2. Search (1).
3. Stop

Algorithm BB1.1

Procedure Search (T)

1. Use Algorithm A.2 to generate one assignment σ^T.

2. SolveLinearProgram (LpSolution, complete).

3. If LpSolution > BestSolution then

   if solution complete then

      set BestSolution = LpSolution

   else

      if T < time horizon then

         Search (T + 1)

4. If level T of the search tree is not empty then

   go to 1

5. Return

4.3.4 Branch and bound Algorithm Version 2

The evaluation step of the first version of the algorithm consists of the solution of an LP for each subproblem considered. However, careful consideration of the underlying structure of the MILP problem results in rules under which a number of subproblems can be evaluated without solving the LP, thus increasing the efficiency of the algorithm.

In the batch processing system we are dealing with, the performance of a certain configuration is associated with the efficient utilization of the corresponding equipment items. Consider two assignments differing only in that some of the units which are processing tasks in one of them stay idle in the other, over the same time periods. Then, it is intuitively obvious that the first one will perform better than the second, provided that there are no units utilization costs. It is

possible to exploit this dominance property within the branch and bound; if the first assignment is evaluated, then an upper bound on the performance of the second is already available without the need to solve an LP problem. Thus some branches of the tree are implicitly enumerated and the search is curtailed.

Before we proceed to state formally the dominance property of the system, we will establish a theorem on which the mathematical proof of the property is based and we will define the subassignment relationship.

THEOREM 1.

Consider two problems of the form:

$$\max\ g_1(x) = f(x) - a_1$$

subject to (Problem 1)

$$\underline{x} \in \Omega_1$$

and

$$\max\ g_2(x) = f(x) - a_2,$$

subject to (Problem 2)

$$\underline{x} \in \Omega_2$$

where a_1, a_2 are constant terms

If

Problems 1 and 2 are bounded; and

the feasibility domains satisfy the relationship

$$\Omega_2 \quad \Omega_1, \quad (\text{ie. } \forall\ x \in \Omega_2,\ x \in \Omega_1),$$

then

$$g_2(x_2^*) \leq g_1(x_1^*) - a_2 + a_1,$$

where x_1^* and x_2^* are the optimal solutions of problems 1 and 2 respectively.

Proof of Theorem 1

The optimal solutions x_1^*, x_2^* do not depend on the constant terms a_1, a_2 appearing in the objective functions. Therefore, x_1^* and x_2^* are also the optimal solutions to the problems

max f(x) , subject to $\underline{x} \in \Omega_1$ (Problem 1.1); and

max f(x) , subject to $\underline{x} \in \Omega_2$ (Problem 2.1)

Since $\Omega_2 \subseteq \Omega_1$, ie. the feasible region of problem 2.1 is contained within the feasible region of Problem 1.1, it follows that:

$f(x_2^*) \leq f(x_1^*)$, or

$g_2(x_2^*) + a_2 \leq g_1(x_1^*) + a_1$, or

$g_2(x_2^*) \leq g_1(x_1^*) + a_1 - a_2$

Corollary

If $\tilde{g}_1^*$ is an upper bound on the optimal value of the objective function of Problem 1, ie $\tilde{g}_1^* \geq g_1(x_1^*)$, then

$g_2(x_2^*) \leq \tilde{g}_1^* + a_1 - a_2$

An assignment σ_2^T is called subassignment of σ_1^T ($\sigma_2^T \subseteq \sigma_1^T$) if:

1. $W_{ijt}^2 \leq W_{ijt}^1$ $\forall$ j, $i \in I_j$, $t \leq T$; and

2. if $W_{ijt}^2 < W_{ijt}^1$, then $t + p_i \leq T$

In assignments σ_1^T and σ_2^T all W_{ijt} variables have been fixed to zero or one for $t \leq T \leq H$. The above definition implies that if a unit is idle in assignment σ_1^T, then it must also stay idle in σ_2^T over the same time periods. On the other hand, if a unit is idle in assignment σ_2^T, while the same unit is processing a task at the same time in assignment σ_1^T, then the processing of the task must not be extended beyond time T.

The subassingnment relationship defined above can better be illustrated with an example. Consider Figure 4.1 which depicts the Gantt charts of two pairs of assignments (σ_1^T, σ_2^T) and (σ_3^T, σ_4^T). (General description of this form of graphical representation has been given in Section 3.5.2).

In Figure 4.1 assignment σ_2^T is a subassignment of σ_1^T,

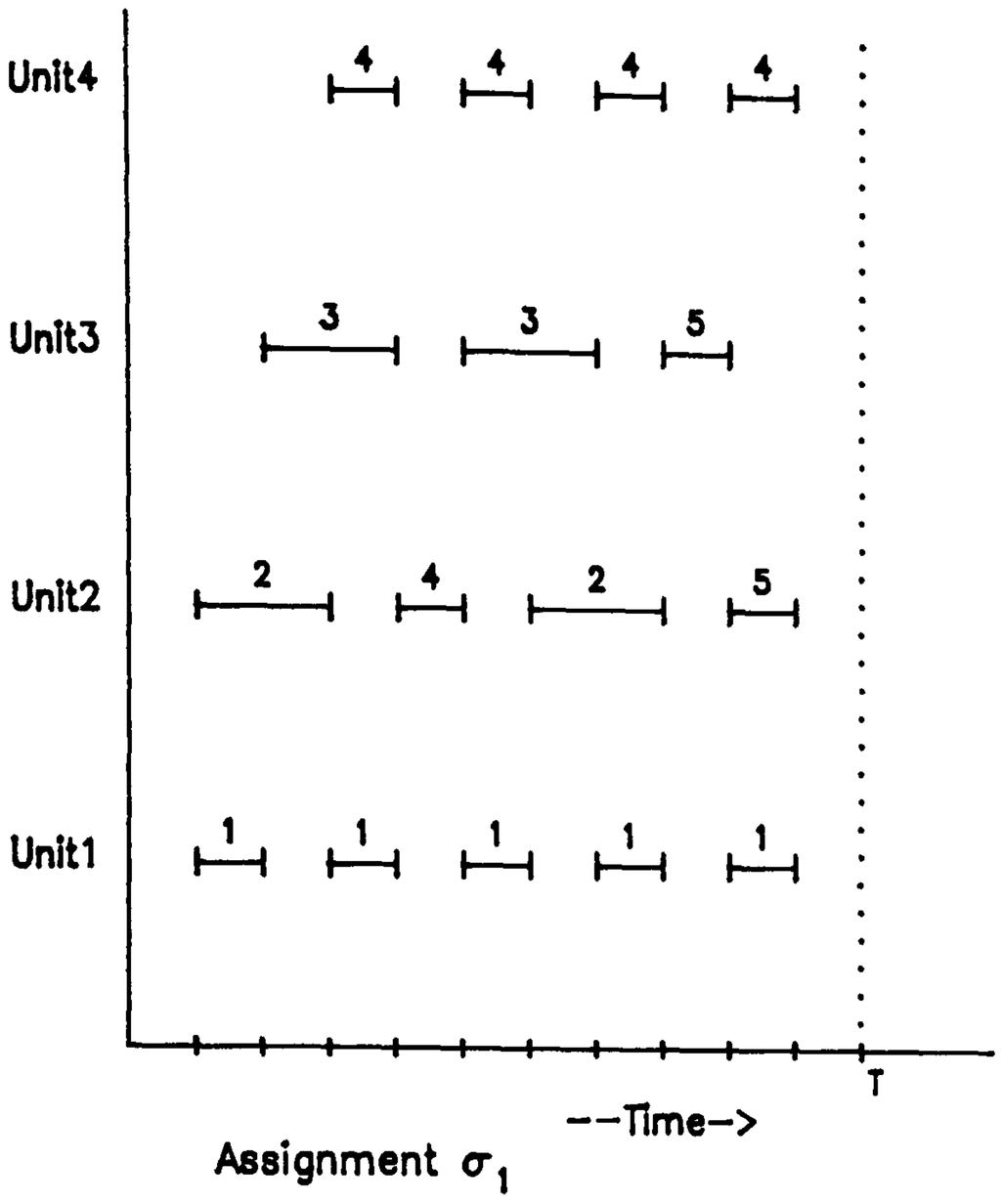
since

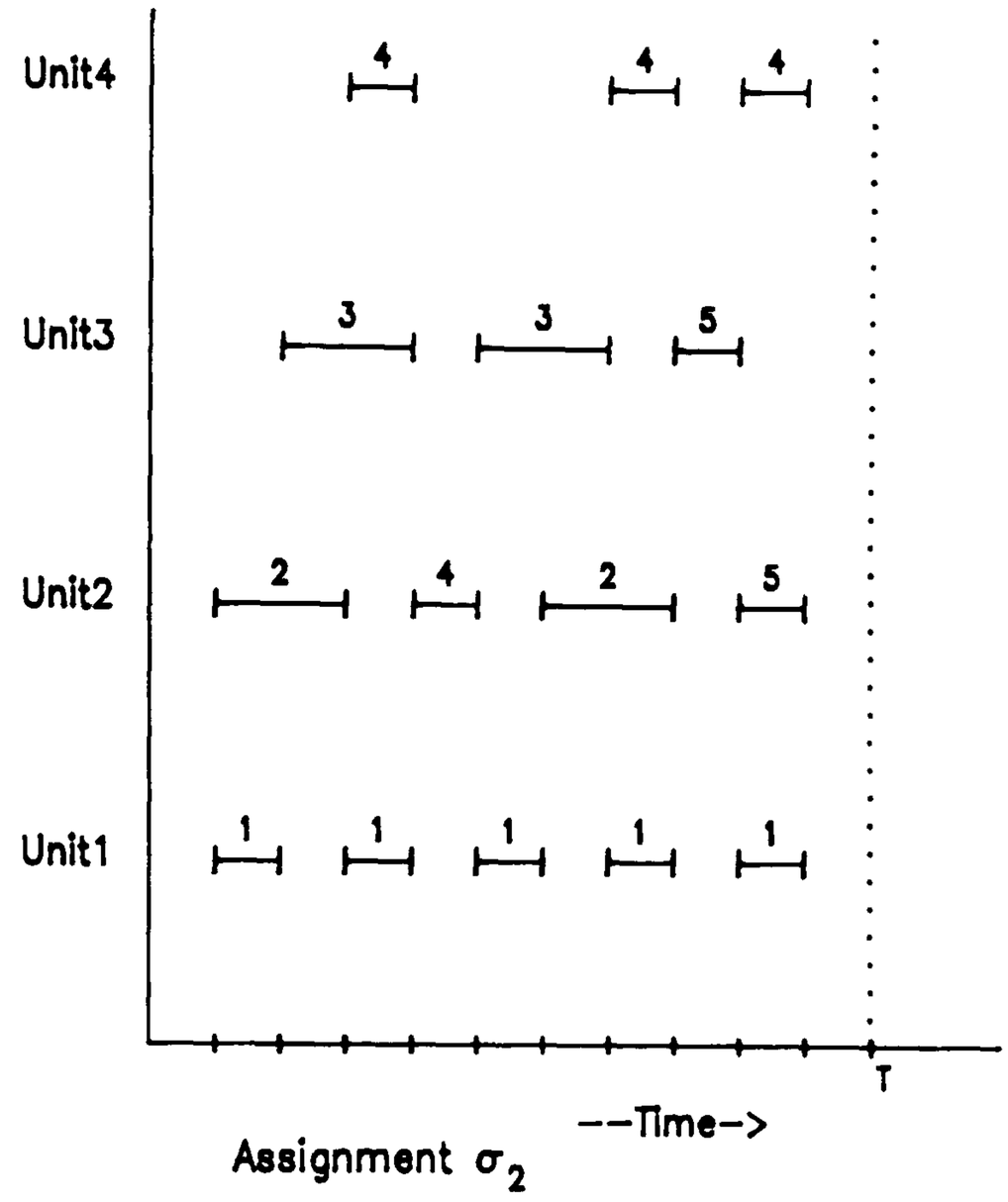
$$W_{ijt}^2 \leq W_{ijt}^1 \quad \text{for all } j,\ i \in I_j,\ t \leq T \qquad \text{(condition 1)}$$

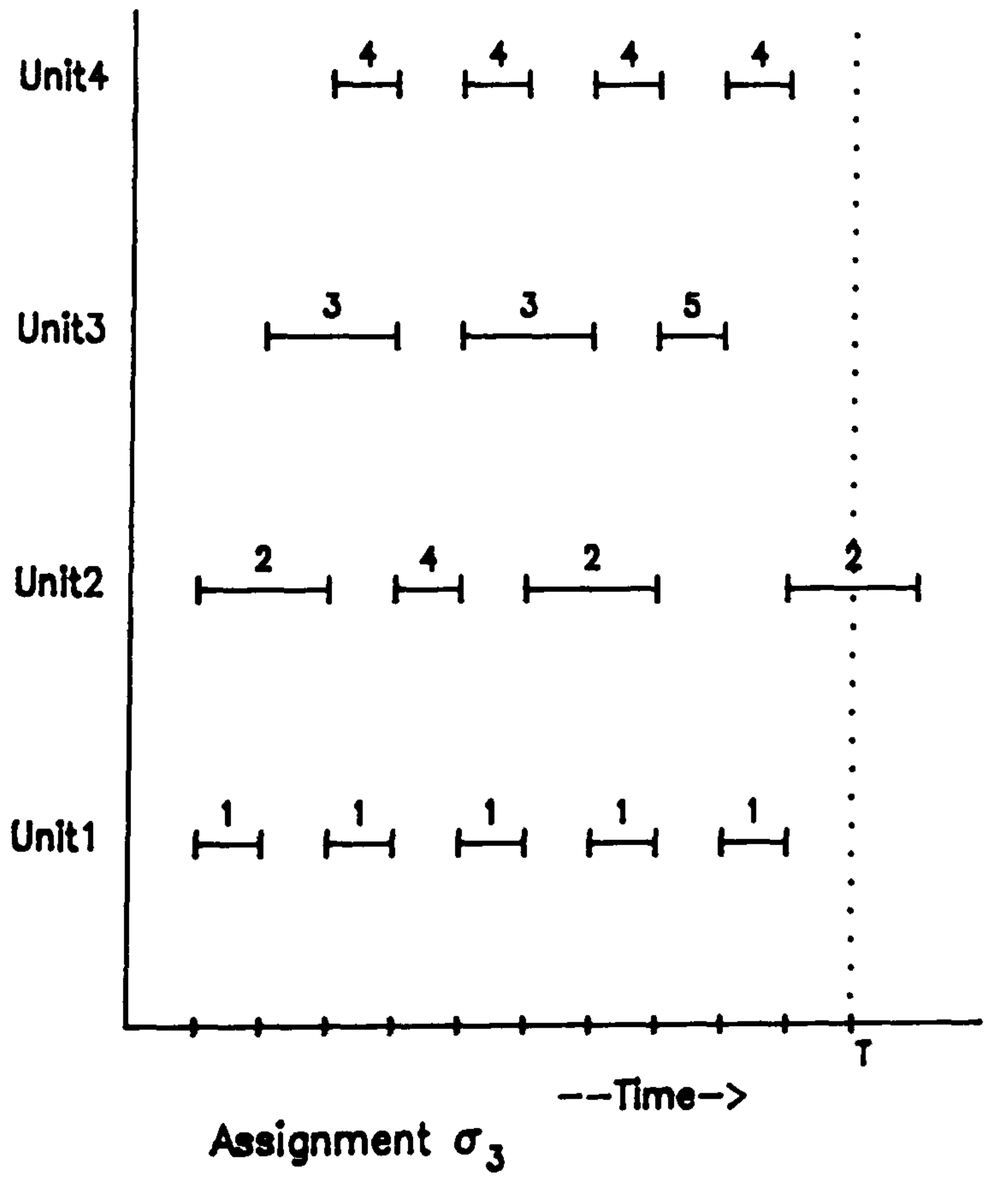
$$W_{445}^2 = 0,\ W_{445}^1 = 1,\ t + p_i = 5 + 1 < T \qquad \text{(condition 2)}$$

In contrast, assignment σ_4^T is not a subassignment of σ_3^T since condition 2 of the above definition is not satisfied

$$W_{22,T-1}^2 = 0,\ W_{22,T-1}^1 = 1 \text{ but } t + p_2 = T - 1 + 2 = T + 1 > T$$







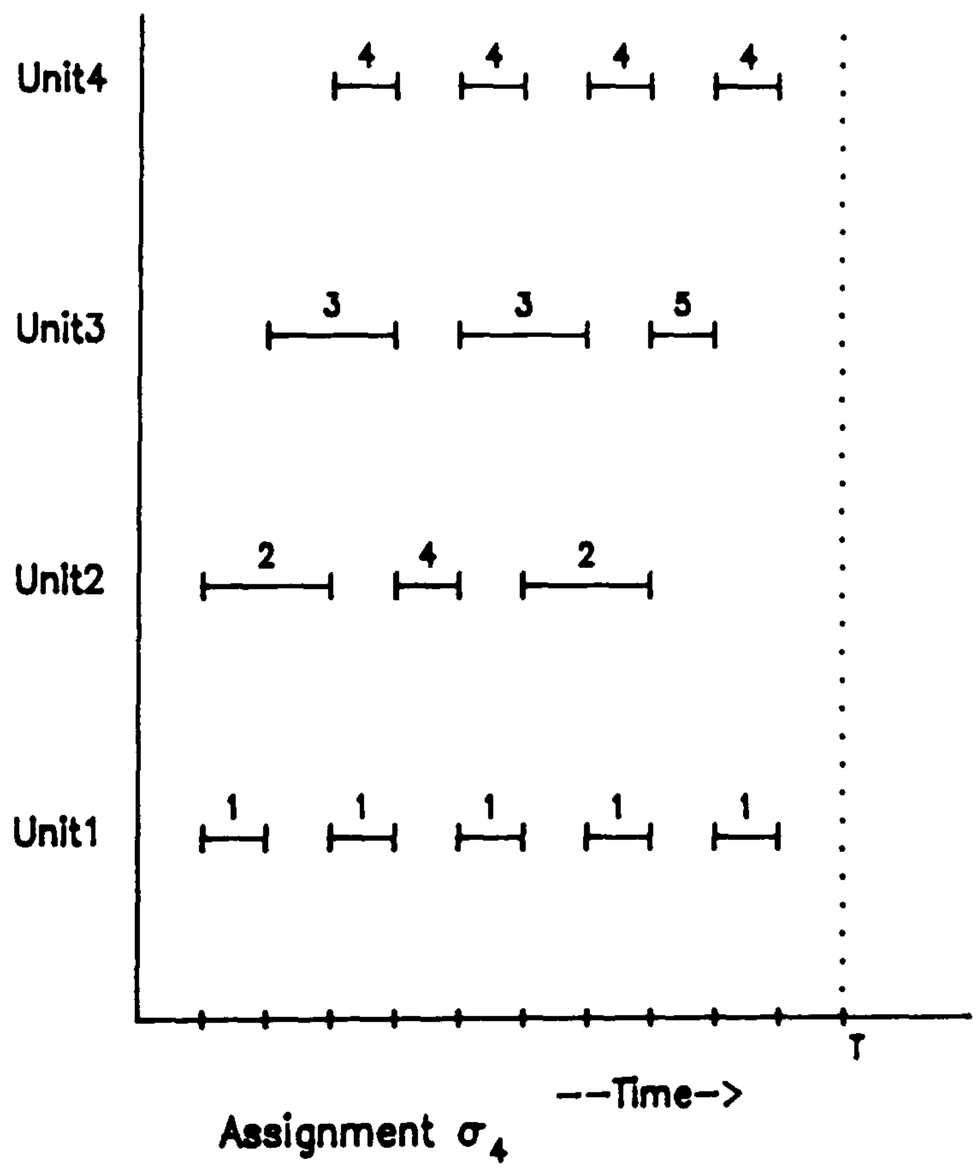


Figure 4.1 Illustration of subassignment definition

Property I

Let P_{σ_1} and P_{σ_2} denote two partial MILP subproblems which appear at the same level of the search tree, corresponding to time T. In P_{σ_1} and P_{σ_2} all W_{ijt} variables for $t \leq T$ have been assigned fixed zero/one values, while W_{ijt} for $T < t \leq H$ are continuous variables between zero an one, ie. $0 \leq W_{ijt} \leq 1$, $T < t \leq H$.

Let σ_1^T and σ_2^T denote the two assignments, corresponding to subproblems P_{σ_1} and P_{σ_2} respectively.

If:

1. An upper bound $\tilde{z}_1$ on the solution of subproblem P_{σ_1} has been calculated; and

2. $\sigma_2{}^T \subseteq \sigma_1{}^T$,

then

$$\bar{z}_2 \leq \tilde{z}_1 + a_1 - a_2,$$

where
$$a_1 - a_2 = \sum_{j=1}^{NUnits} \sum_{i \in I_j} c_{ij} \sum_{t=1}^{T} W^1_{ijt}$$

for j, $i \in I_j$, $t \leq T$ such that $W^2_{ijt} < W^1_{ijt}$

Proof of Property I

The objective function is of the form

$$z = y\ (L_{ijt}) - \phi(W_{ijt})$$

The term $\phi(W_{ijt})$ expresses the units utilization cost. Since all W_{ijt} are fixed for $t \leq T$, $\phi(W_{ijt})$ can be partitioned into a constant term

$$a = \phi(W_{ijt}),\ t \leq T$$

and

$$\phi(W_{ijt}),\ t > T$$

Hence

$$z = y\ (L_{ijt}) - \phi\ (W_{ijt})\big|_{t>T} - a, \text{ or } z = f(L,W) - a$$

By definition, σ_1^T and σ_2^T both satisfy constraints (3.1). The W_{ijt} otherwise appear only on RHS of constraints (3.2) and in view of conditions (1) and (2) in the definition of subassignment, we have

$$W^2_{ijt} \leq W^1_{ijt},\ \forall\ i, j \in K_i,\ t \leq T$$

Thus, the feasibility domain of subproblem P_{σ_2} is contained within the feasibility domain of subproblem P_{σ_1}.

It is important to notice that the variables of subproblems P_{σ_1} and P_{σ_2} defined for periods $t > T$ are not affected by fixing all W_{ijt}, $t \leq T$, since $t + p_i \leq T$ for $W^2_{ijt} < W^1_{ijt}$. Otherwise, variables $W_{ijt'}$ and $L_{ijt'}$, $t < t' \leq t + p_i$, $t + p_i > T$, which would be forced to zero in σ_1^T, could

take non-zero values in σ_2^T, thus increasing the feasibility region of subproblem P_{σ_2}.

Since the objective function of subproblems 1 and 2 is of the same form as the functions $g_1(x)$ and $g_2(x)$ of Theorem 1 and condition 1 of the theorem is satisfied, it is concluded that

$$\bar{z}_2 \leq \bar{z}_1 + a_1 - a_2$$

In this case

$$a_1 = \sum_{j=1}^{NUnits} \sum_{i \in I_j} c_{ij} \sum_{t=1}^{T} W_{ijt}^{1}$$

$$a_2 = \sum_{j=1}^{NUnits} \sum_{i \in I_j} c_{ij} \sum_{t=1}^{T} W_{ijt}^{2}$$

and, in view of condition 1 of the subassignment definition

$$\left(\text{i.e. } W_{ijt}^{2} \leq W_{ijt}^{1}\right),$$

$$a_1 - a_2 = \sum_{j'} \sum_{i' \in I_{j'}} c_{i'j'} \sum_{t'} W_{i'j't'}^{1}$$

for j', $i' \in I_{j'}$, $t' \leq T$ such that

$$W_{i'j't'}^{2} < W_{i'j't'}^{1}$$

Thus,

$$\bar{z}_2 \leq \bar{z}_1 + \sum_{j'} \sum_{i' \in I_{j'}} c_{i'j'} \sum_{t'} W_{i'j't'}^{1} \qquad (4.8)$$

If the objective function does not involve units utilization costs, then

$$z = y(L_{ijt})$$

and $a_1 = a_2 = 0$,

thus

$$\bar{z}_2 \leq \tilde{z}_1$$

Property I has the following implication:

Consider that at a level T of the search tree a set of subassignments σ_i^T satisfy the relationship $\sigma_i^T \subseteq \sigma_1^T$. If all the branches emanating from node P_{σ_1} have been enumerated (implicitly or explicitly), thus calculating an upper bound $\tilde{z}_1$ on the solution of P_{σ_1}, then the partial MILPs corresponding to all subproblems P_{σ_i} need not be solved, since an upper bound to their solution can be obtained by using relation (4.8).

Furthermore, if the objective function does not involve units utilization costs and subproblem P_{σ_1} has been discarded, then all subproblems P_{σ_i} need not be considered any further in the search of the optimum, since $\bar{z}_i \leq \tilde{z}_1$. Consequently, all the branches emanating from nodes P_{σ_i} are fathomed automatically.

By exploiting Property I within the branching tree, a number of branches are implicitly enumerated and the solution of a large number of LPs is avoided, thus increasing the efficiency of the algorithm. It is worth noting that this increase of the efficiency has been achieved without compromising the optimality of the solution obtained.

In the second version of the algorithm the type of search is again depth-first, as in version 1. Essentially the only difference between the first and the second version is in Step 2 of Algorithm A.1.1, in the evaluation of the subproblem.

In order to utilize Property I the condition $\sigma_i \subseteq \sigma_1$ must be detected. An efficient technique found for carrying out the dominance check is the following: for each time period T a comparison-list (T) is defined, storing all assignments generated at time T. The records kept for each stored assignment σ_1^T include the identifiers of tasks to which the units have been allocated and an associated value of the objective function estimate (ObjFunEst).

This value is the upper bound on the optimal solution of the corresponding MILP problem P_{σ_1}, ie. is the term $\bar{z}_1$, as it appears in relation (4.8). In order to calculate it, a second argument (ofest) has been introduced in procedure Search. Since all variable W_{ijt} have been fixed for $t \leq T$ in subproblem P_{σ_1}, ofest is the best completion of the branch. More specifically, having enumerated all branches emanating from P_{σ_1}, ofest is either the best complete solution found amongst them, or, if no complete solution is encountered, ie. the branches are cut, is the best amongst the upper bounds of these branches.

If after comparison an assignment σ_2^T is found to be subassignment of σ_1^T, then subproblem P_{σ_2} is not solved and it is given an upper bound $\bar{z}_2$ (LpEst) obtained from relation (4.8).

Version 2 of the algorithm is summarized below:

Algorithm BB2

1. Construct a trial solution, using Algorithm A0.

   Set BestSolution = $\bar{z}_{init}$

2. For T = 1 to time horizon do

   Set comparison-list(T) = empty

3. Search (1,ofest).

4. STOP

Algorithm BB2.1

Procedure Search (T,ofest)

1. Set ofest = 0.0

2. Use Algorithm A.2 to generate one assignment σ^T

3. CompareAssignment (σ^T, subassignment, LpEst)

   If subassignment then

      Set Ubound = LpEst

      Set solution complete = false

   else

      SolveLinearProgram (LpSolution, complete)

      Set Ubound = LpSolution

4. If Ubound > BestSolution then

      If solution complete then

      Set BestSolution = Ubound

      Store σ^T in comparison-list(T) with associated ObjFunEst=Ubound

      Set ofest = max(ofest, Ubound).

   else

```
if T < time horizon  then

   Search (T+1, ofnew)

   Store σ^T in comparison-list(T) with associated ObjFunEst =

   ofnew

   Set ofest = max(ofest, ofnew)

   else

      Store σ^T in comparison-list(T) with associated ObjFunEst=Ubound

      ofest = max (ofest,Ubound)

5.  If level T of the tree is not empty then

      Go to 2

6.  Return
```

4.3.5 Branch and Bound algorithm Version 3

This version of the algorithm differs from the second one in the order in which the nodes of the tree are explored. Instead of branching from the newest generated node, this version employs a combination of depth-first and breadth-first search. Thus, referring to the Algorithm A.1.1, (Section 4.3.2), Step 1 generates all the subproblems with the same parent node. In Step 2 the subproblems are evaluated as in version 2 and the promising ones are stored in assignment-lists defined for each time period. The third Step arranges the subproblems stored in the assignment-list in decreasing value of their upper bounds and Step 4 explores the subproblems in this order.

The structure of the third version of the algorithm is as follows.

Algorithn BB3

Same as Algorithm BB2 (Section 4.3.4)

Algorithm BB3.1

Procedure Search (T, ofest).

1. Set assignment-list (T) = empty
2. Set ofest = 0.0
3. Use Algorithm A.2 to generate all assignments σ_i^T
4. For all σ_i^T do

Compare assignment (σ_i^T, subassignment,LpEst)

If subassignment then

set Ubound = LpEst

set solution complete = false

else

SolveLinearProgram (LpSolution complete)

set Ubound = LpSolution

If Ubound > BestSolution then

if solution complete then

set BestSolution = Ubound

store σ_i^T in comparison-list(T) with associated

ObjFunEst=Ubound

set ofest = max(ofest, Ubound)

else

store σ_i^T and corresponding Ubound in assignment-list(T)

else

store σ_i^T in comparison-list(T) with associated ObjFunEst = Ubound

set ofest = max (ofest, Ubound).

5. Order assignments in assignment-list (T) in non-decreasing order of Ubound

6. While assignment-list (T) is not empty and Ubound > BestSolution do

If T < time horizon then

Search (T + 1, ofnew)

store σ_i^T in comparison-list (T) with associated ObjFunEst=ofnew

set ofest = max(ofest, ofnew)

7. Return

Notice that if the Ubound of the first assignment stored in assignment-list(T) is less than the best solution yet found, the algorithm proceeds directly to Step 7.

4.4 Concluding Remarks

Branch and bound methods are one of the most effective tools in solving MILP problems, especially when the structure of the particular problem is exploited.

We have developed a schedule generating algorithm employing the branch and bound method to obtain the optimal solution of the MILP model which describes our problem. Starting from the same basic structure, three versions of the algorithm have been developed, differing in the way the nodes of the tree are evaluated and in the order they are explored.

In the basic form of the algorithm (version 1) the branching is carried out upon the newest generated node and the fathoming of non promising branches is based on the optimal solution of the corresponding LP.

In a second version, the search is again depth-first and the algorithm employs a dominance property to eliminate certain partial assignments from further consideration. The same property which stems from physical insight of the system is applied in order to estimate bounds of various subproblems of the tree, without solving the LP. The efficiency of the algorithm is increased substantially and the optimality of the solution obtained is still guaranteed.

The difference between the second and the third version of the algorithm is the order in which the nodes of the tree are scanned. Thus, the third version orders the subproblems which have the same parent node according to the value of their upper bounds and explores the node with the largest bound first. The motivation for developing this version has been the fact that in a general branch and bound procedure the complete solutions encountered by this type of search tend to be close to the optimal, while the early complete solutions obtained by the depth-first search might be relatively poor. However, in our case, the computational efficiency of the third version, although better than that of version 1, is not as good as the efficiency of version 2. The reasons for this are discussed in Chapter 5. Finally, a simple and efficient algorithm has been developed in order to obtain a trial solution of the problem, used to initialize the branch and bound search.

All the versions of the algorithm have been implemented as PASCAL procedures, which call the MINOS package to solve the Linear Programming problems.

Computational experience with the algorithm is discussed in the next chapter.

C H A P T E R F I V E

CASE STUDIES

5.1 Introduction

A number of experiments have been carried out in order to illustrate the application of the proposed solution strategy in batch operating systems of varying structure and complexity.

In this Chapter four examples will be considered, containing different potential problem features. In all cases the data required in order to find the optimal schedule are:

1. the time horizon of interest;
2. process specification, ie. the tasks to be carried out, their precedence constraints and processing times (the recipe);
3. plant specification, ie. the number, size and suitability of the units;
4. stock holding information, ie. the available storage in all the states of the Network;
5. initial inventory, ie. the amount of material stored in the process states at the beginning of the time horizon; and
6. planning and market data and the objective criterion to be optimized.

The problem solution is defined by:

1. the resource allocation over time (ie. the values of the binary variables); and
2. the processed amounts of material and the inventory levels in the plant (ie. the values of the continuous variables).

5.2 Batch System Problem 1

The process is described by the State Task Network of Figure 5.1. The required data are given in Table 5.1.

Table 5.1 Specifications for Problem 1

Time Horizon = 12 hours				
Process Units	Size (lt)	Units Suitability	Tasks processing time (hour)	Units utilization costs c_{ij}
Unit 1	1500	Task 1	1	2.0
Unit 2	1000	Task 2	1	2.0
Unit 3	1000	Task 3	1	2.0
Process States		Storage Capacity (lt)	Initial amount stored(lt)	Product prices/ Feed costs
State 1 (Feed)		unlimited	unlimited	5.0
State 2 (Intermediate)		5000	0.0	0.0
State 3 (Prod1)		unlimited	0.0	10.0
State 4 (Prod2)		unlimited	0.0	8.0
Objective: maximize profit				

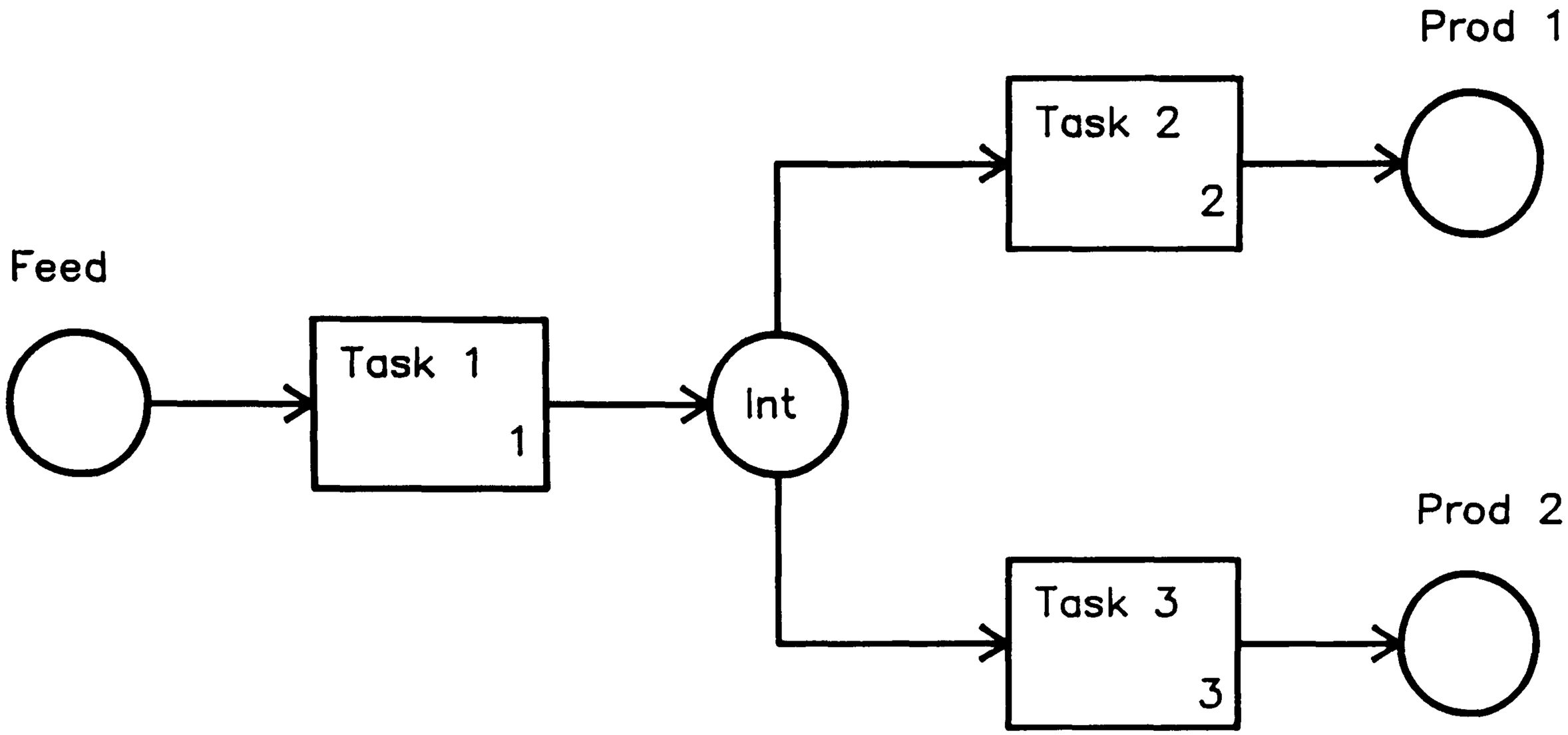


Figure 5.1 State Task Network for Problem 1

The optimal schedule of the above problem is shown in Figure 5.2. The number above each line segment identifies the processing task, whereas the number below it is the amount of material which undergoes the corresponding task.

Since both Tasks 2 and 3 get their inputs from state 2 and the size of Unit 1 is less than the sizes of Units 2 and 3, the limiting stage of the process is Task 1. The second task is executed more frequently than Task 3 because the unit price of Product 1 (10.0) is higher than that of Product 2 (8.00).

The dimensionality of the problem, the value of the objective function at the optimal solution and the number of LPs solved in the branch and bound search are shown in Table 5.2.

Table 5.2 Dimensionality and Results of Problem 1

Number of integer variables	36
Number of continuous variables	88
Number of constraints	120
Number of LPs solved	652
CPU sec on IBM 4341	113
Optimal value of the objective function	64950

The value of the objective function is obtained as follows:

Total amount of Product 1 = 10000

Total amount of Product 2 = 5000

Consumed amount of Feedstock = 15000

$$\sum_{t=1}^{12} W_{11t} = 10$$

$$\sum_{t=1}^{12} W_{22t} = 10$$

$$\sum_{t=1}^{12} W_{33t} = 5$$

Profit = Products Price - Feed Cost - Units Utilization Cost =

10.10000 + 8.5000 - 5.15000 - 2.(10+10+5) = 64950

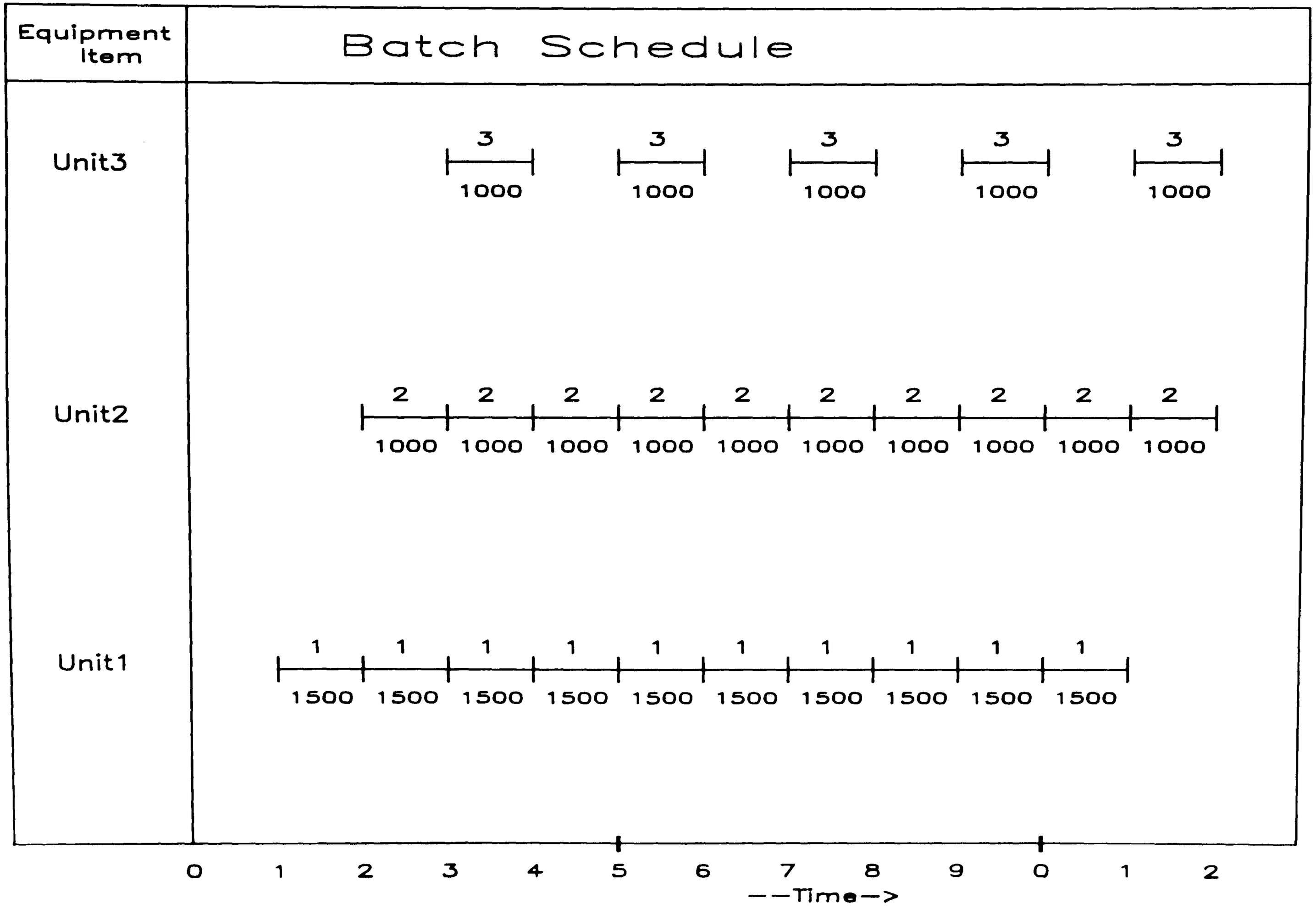


Figure 5.2 Optimal schedule for Problem 1

5.3 Batch System Problem 2

The batch process system described in section 3.2 is considered. The corresponding State Task Network is shown in Figure 5.3 and the specifications of the problem are given in Table 5.3.

Table 5.3 Specifications for Problem 2

Time Horizon = 10 hours				
Process Units	Size(kgr)	Units Suitability	Tasks Processing Time (hrs)	Units Utilization Costs bij
Heater	100	Heat (Task 1)	1	0.2
Reactor 1	50	Reactions 1,2,3	2,2,1	0.32,0.7,0.2
Reactor 2	80	Reactions 1,2,3	2,2,1	0.5,0.5,0.3
Still	200	Separation	1 hr Prod.2 2hrs Int AB	0.0
Process States		Storage Capacity (kgr)	Initial amount stored(kgr)	Product prices/ Feed costs
Feeds A, B, C.		unlimited	unlimited	0.0
Hot A		100	0	0.0
Int AB		200	0	0.0
Int BC		150	0	0.0
Int E		200	0	0.0
Prod 1		unlimited	0	10.0
Prod 2		unlimited	0	10.0
Objective: Maximize profit				

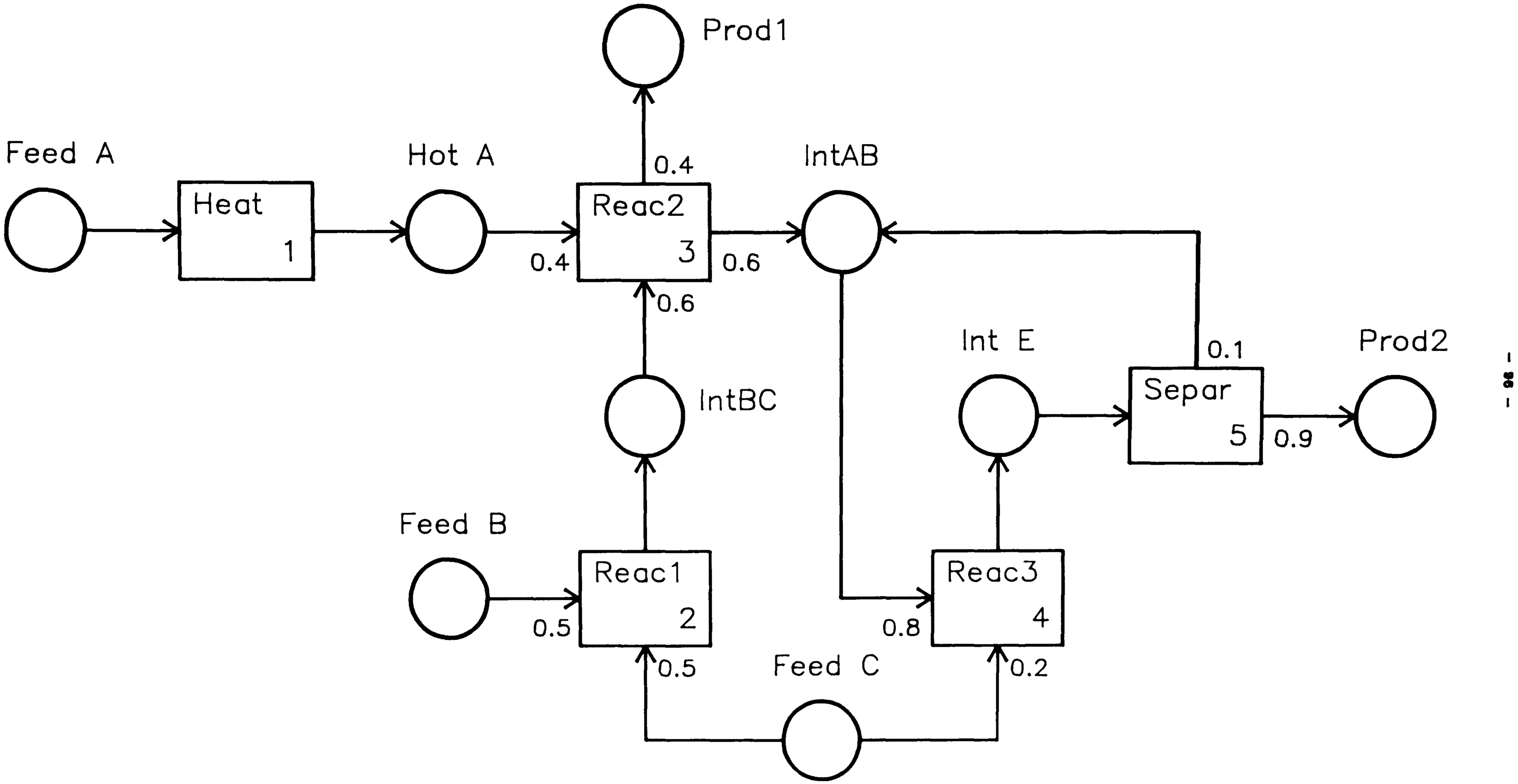


Figure 5.3 State Task Network for Problem 2

The initial solution generating algorithm provides the schedule shown in Figure 5.4 and a corresponding value of the objective function 1257.85.

The optimal schedule of the process and the corresponding inventory levels in some of the states are depicted in Figures 5.5 and 5.6 respectively. The maximum profit is 2082.8.

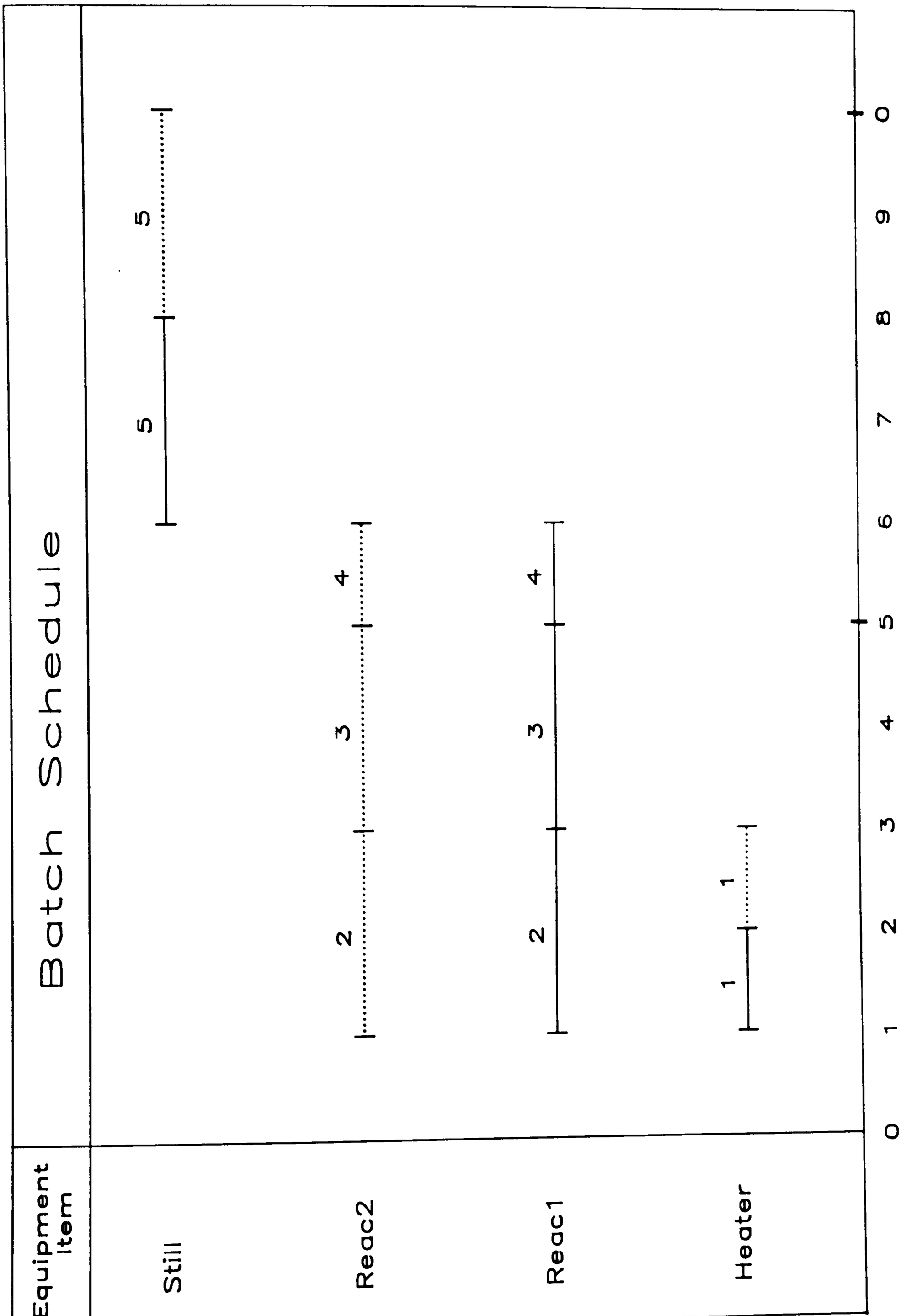


Figure 5.4 Initial solution for Problem 2

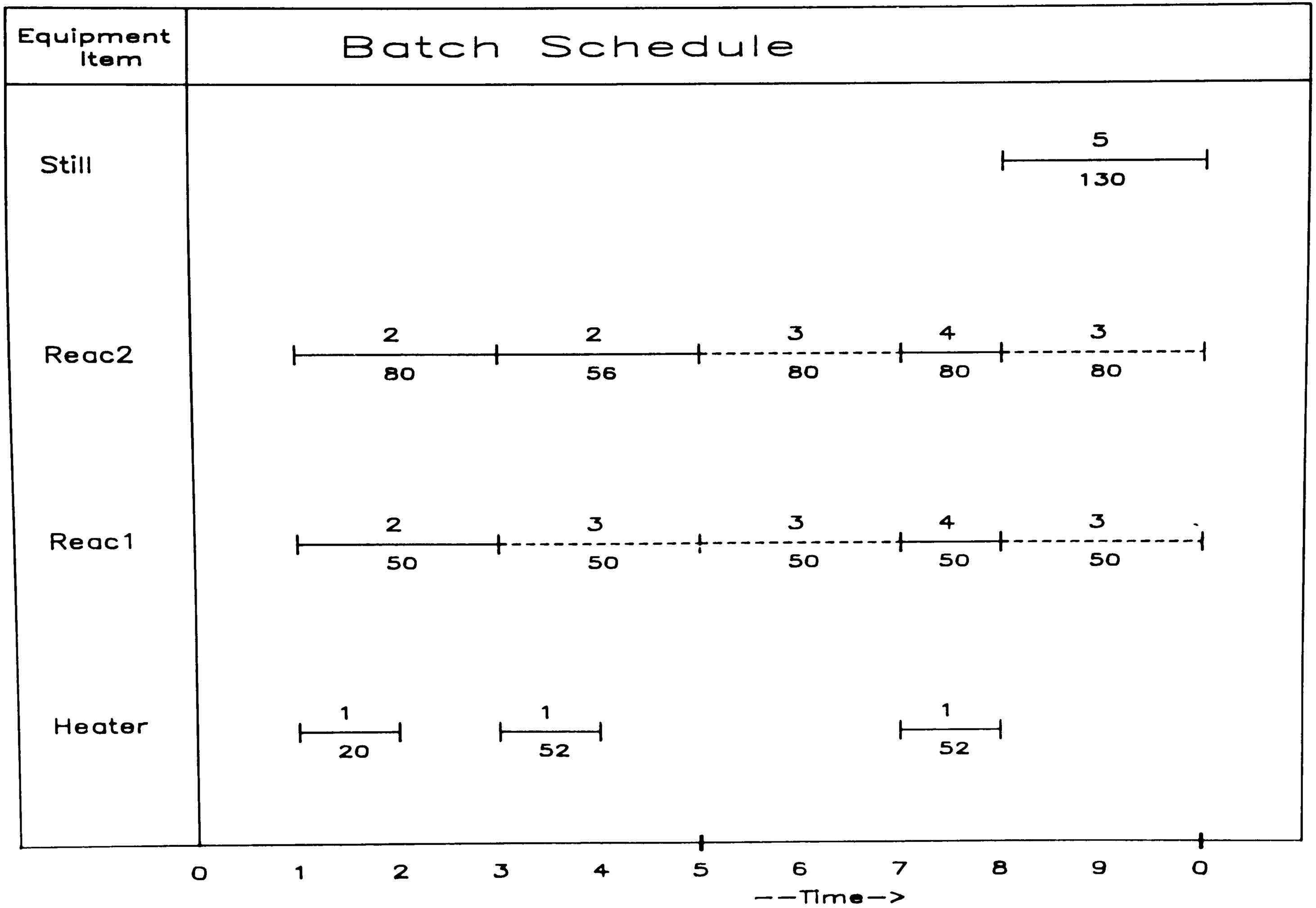
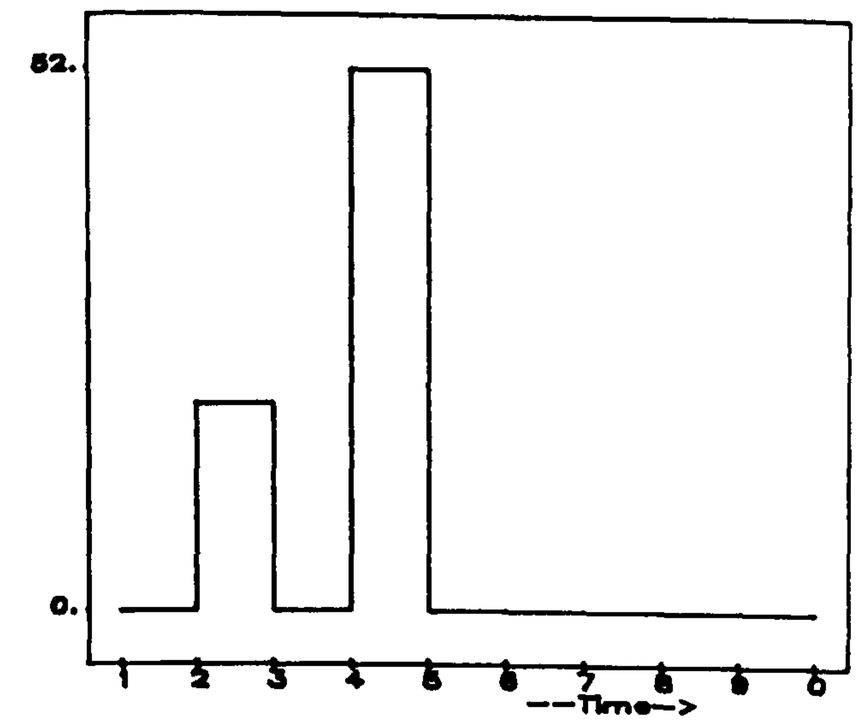
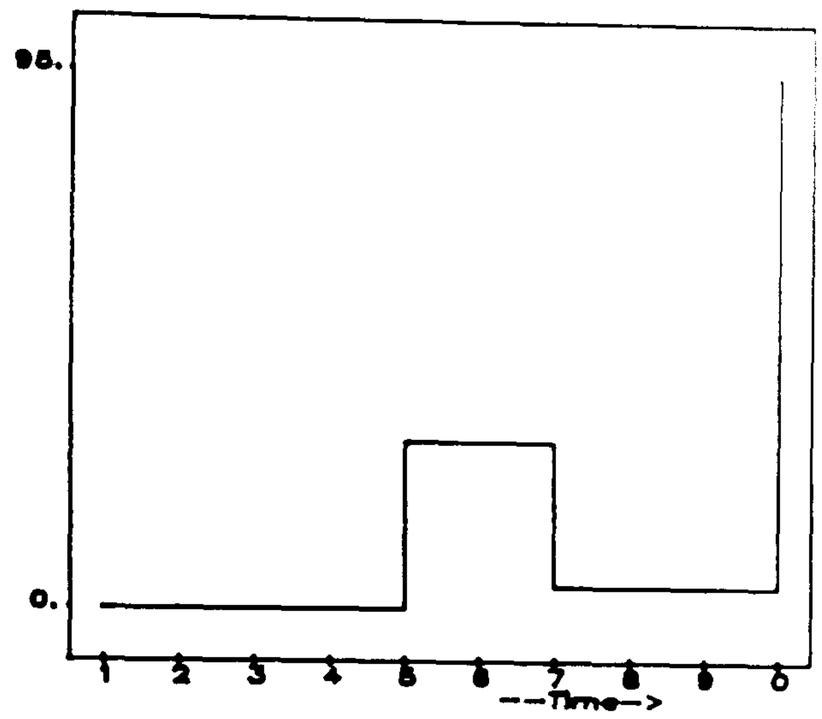


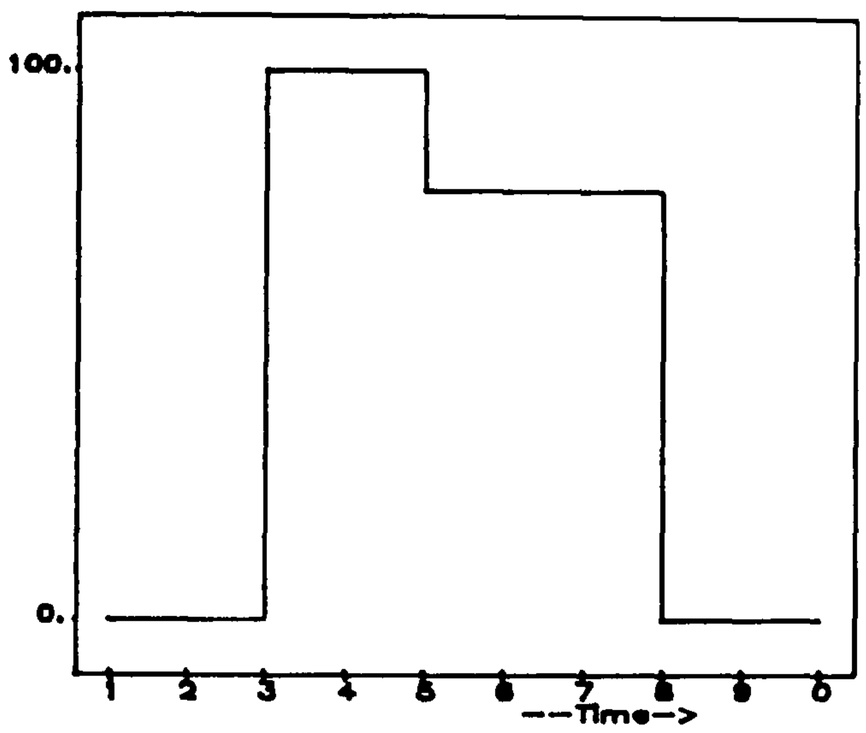
Figure 5.5 Optimal schedule for Problem 2



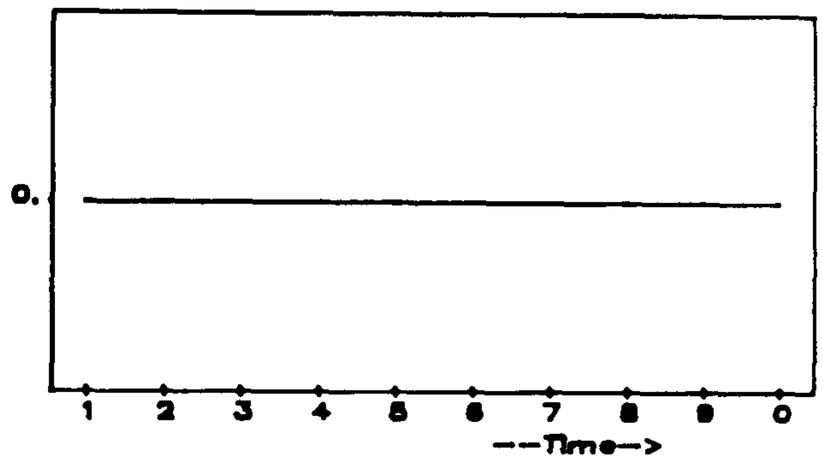
HotA (State4)



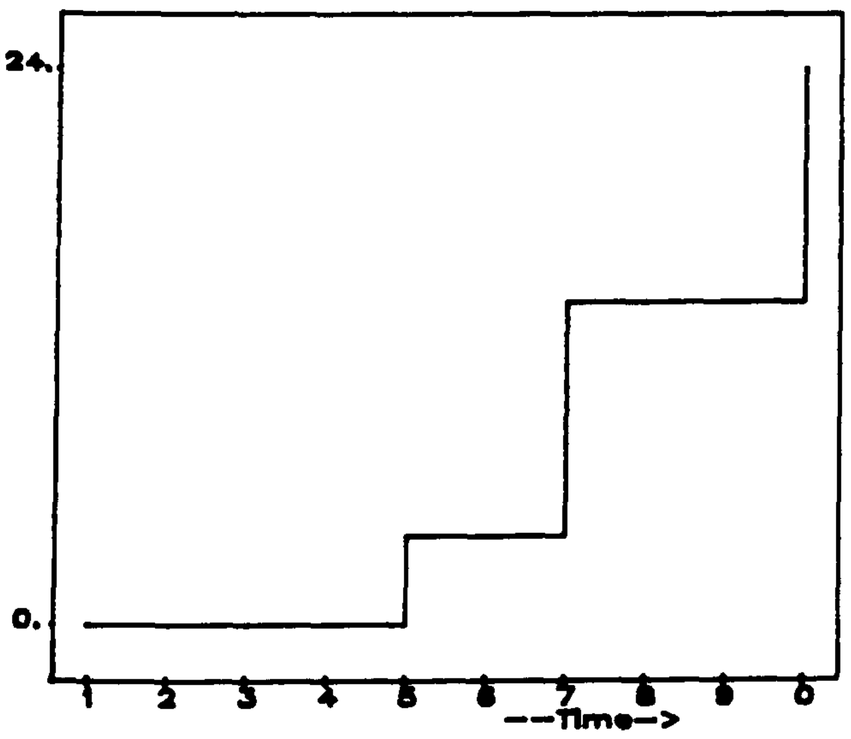
IntAB (State5)



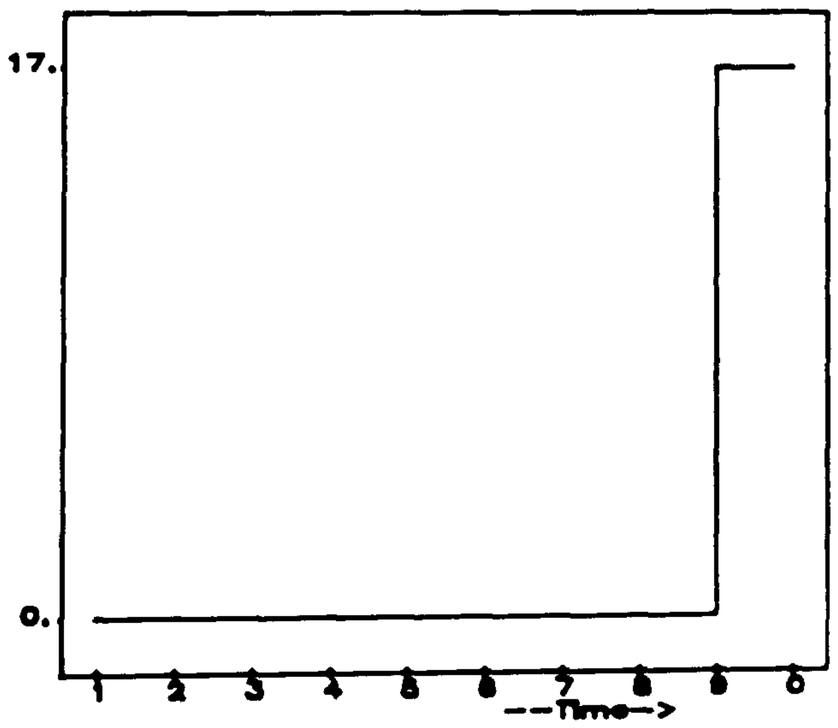
IntBC (State6)



IntE (State7)



Prod1



Prod2

Figure 5.6 Inventory levels for Problem 2

5.3.1 Batch system Problem 2a

In order to investigate the effects of intermediate storage on the performance of the batch system of Problem 2, the same problem has been solved, except that we consider

$$C_{IntBC} = 0 \text{ (instead of 150 kgr)}$$

The optimal schedule of the process in this case and the inventory levels are shown in Figures 5.7 and 5.8 respectively. The optimal profit is 1833.81.

As expected the performance of the plant is worse than before, since the lack of intermediate storage results in less efficient utilization of the resources in the plant.

Task 2, 3 and 4 can be performed in both Reactors 1 and 2. In the optimal solution the unit where each of these tasks is carried out, as well as the amount of material undergoing each task in each unit, are determined in such a way that the units operating cost is the minimum possible. For example, 83.33 kg start undergoing Task 3 at time 8. Since $b_{32} = 0.7$ and $b_{33} = 0.5$, Reactor 2 processes 80 kg and the remaining 3.33 kgr are processed by Reactor 1.

Problem 2a has been solved by using all three versions of the branch and bound algorithm described in Chapter 4. The computational requirements of a branch and bound technique depend on the number of linear programming problems which have to be solved in the course of the search. In this context, version 2 is the most efficient, as is demonstrated in Table 5.4, in which the number of LPs solved in each version to obtain the optimal solution is displayed.

The various characteristics of Problems 2 and 2a are shown in Table 5.4.

Table 5.4 Dimensionality and Results of Problems 2 and 2a

	Problem 2	Problem 2a
Number of integer variables	80	80
Number of continuous variables	179	179
Number of constraints	250	250
Optimal solution	2082.8	1833.81
Number of LPs solved	1466	648
CPU sec on VAX 8600	908	343

Problem 2a	LPs solved	CPU sec on VAX 8600
Algorithm BB2	648	343
Algorithm BB1	17388	7232.6
Algorithm BB3	6917	3223.5

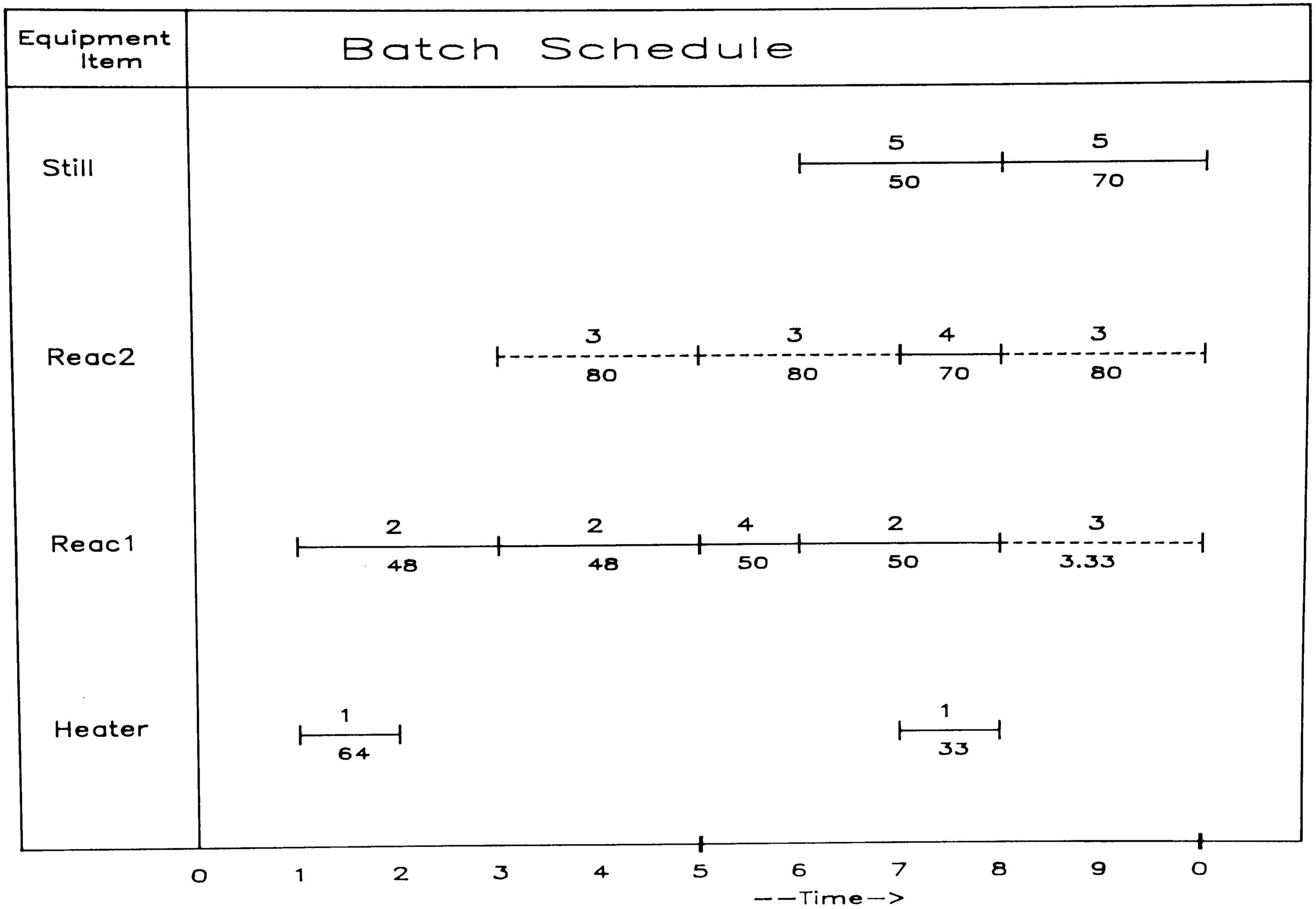


Figure 5.7 Optimal schedule for Problem 2a

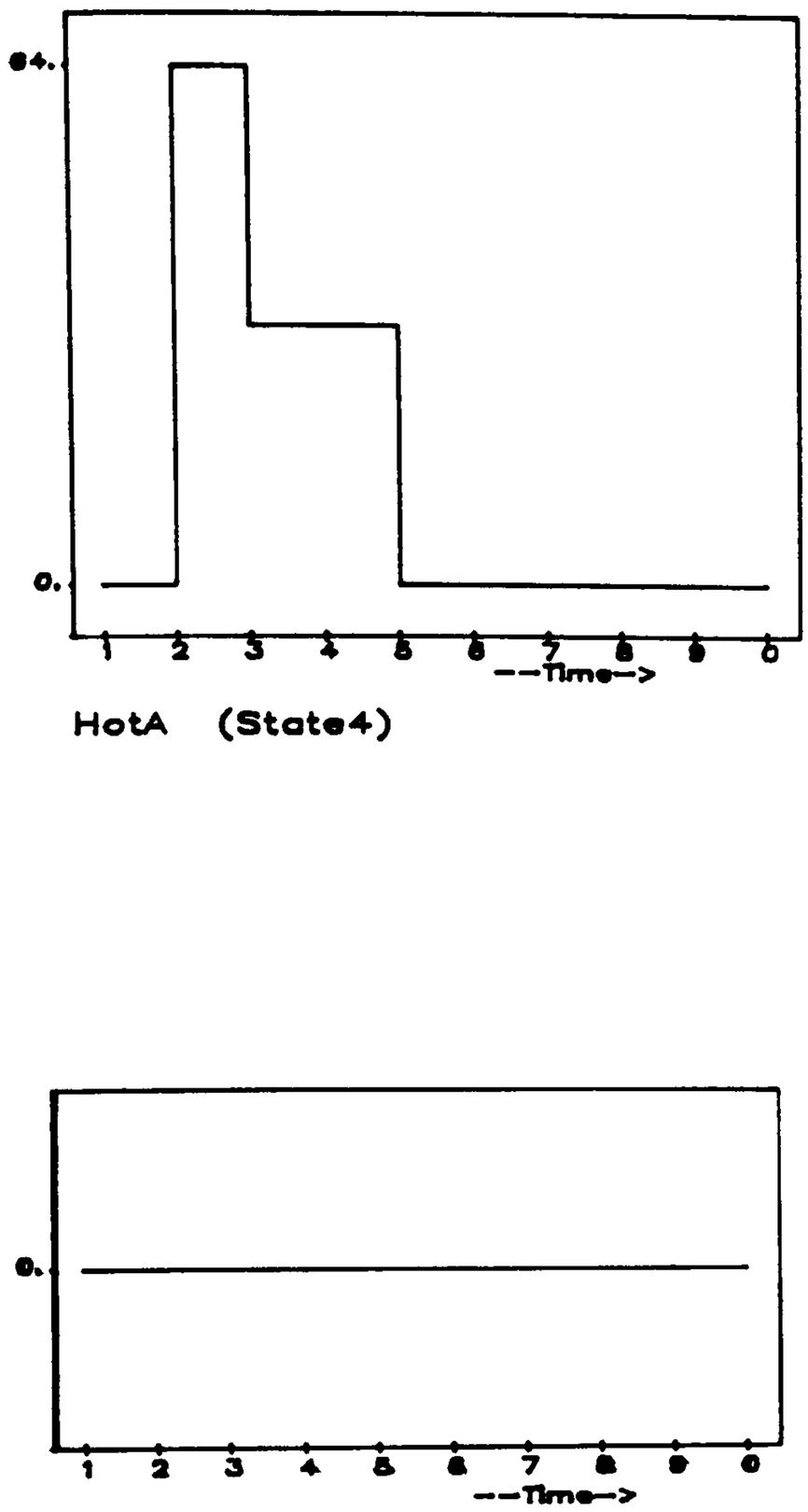


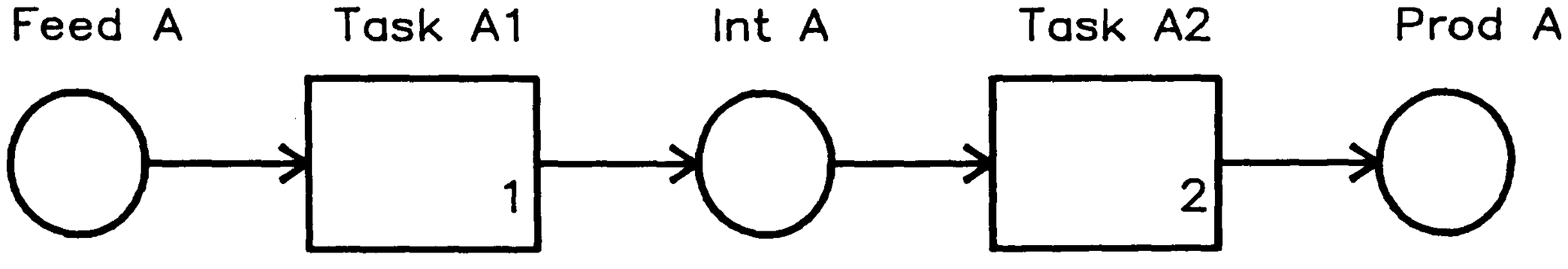
Figure 5.8 Inventory levels for Problem 2a

5.4 Batch system Problem 3

Three products are to be produced using three feedstocks, following the processing scheme of the STN shown in Figure 5.9. The products do not share common intermediates. The specifications of the problem are given in Table 5.5.

Table 5.5 Specifications for Problem 3

Time Horizon = 8 hours			
Process Units	Size(kgr)	Units Suitability	Tasks processing Time (hour)
Unit 1	2029	Task A1	2
Unit 2	1690	Task A2,C1	1,1
Unit 3	720	Task B1,C2	1,2
Unit 4	929	Task B2	1
Process States		Storage Capacity (kgr)	Initial amount stored(kgr)
Feeds A, B, C		unlimited	unlimited
Int A		2500	0.0
Int B		0	0.0
Int C		0	0.0
Products A, B, C		0	0.0
Objective:maximize total production $=(S_{P_A} + S_{P_B} + S_{P_C})_8$			



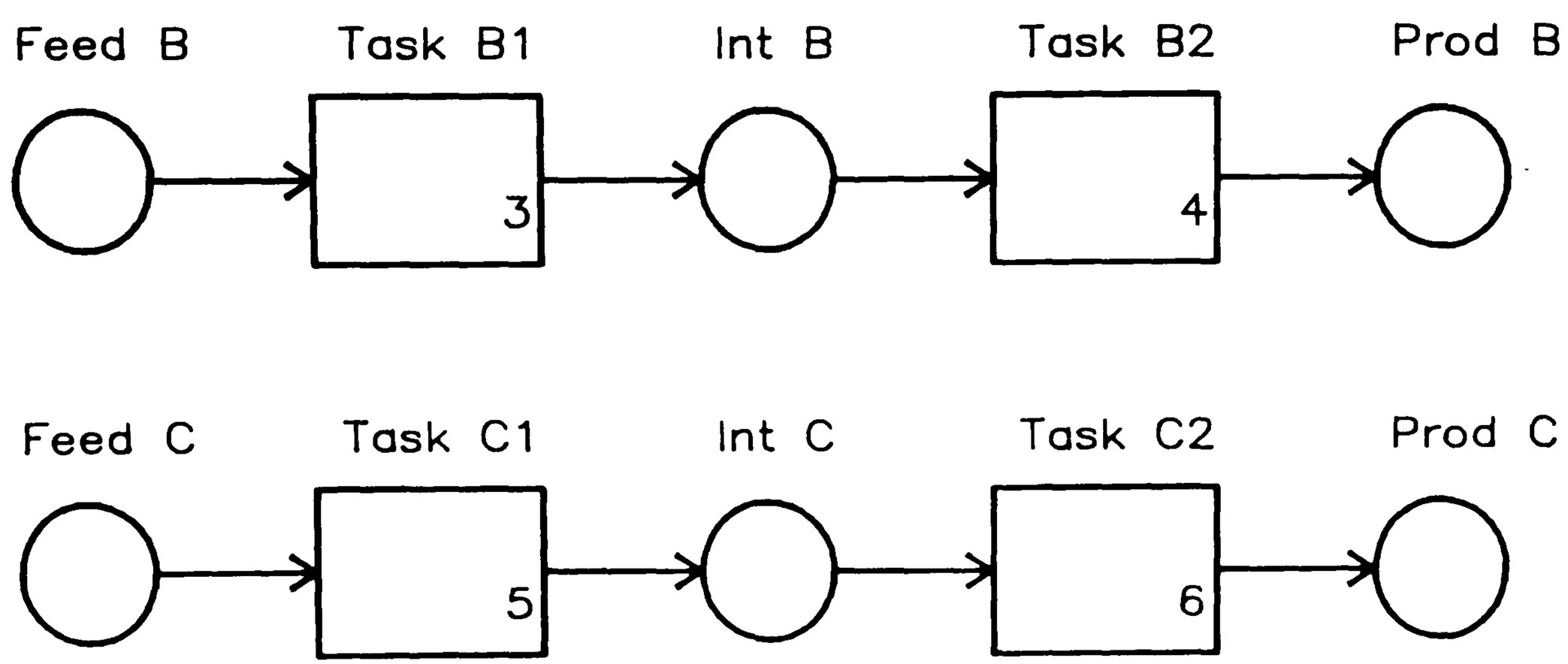


Figure 5.9 State Task Network for Problem 3

The optimal schedule of the process is shown in Figure 5.10 and the dimensionality of the problem as well as the optimal production in Table 5.6.

Table 5.6 Dimensionality and Results for Problem 3

Number of integer vars.	48
Number of cont. vars	129
Number of constraints	168
Optimal production	11494
CPU sec on VAX 8600	688

In the optimal schedule of the problem, Product B is produced more frequently than Product C. This happens because both Tasks B1 and C2 can be performed by Unit 4 only and Task C2 is longer than B1. As the objective function is the production maximization and Unit 3 is available to perform Task B2 every hour, the performance of Task B1 is more advantageous than that of Task C2.

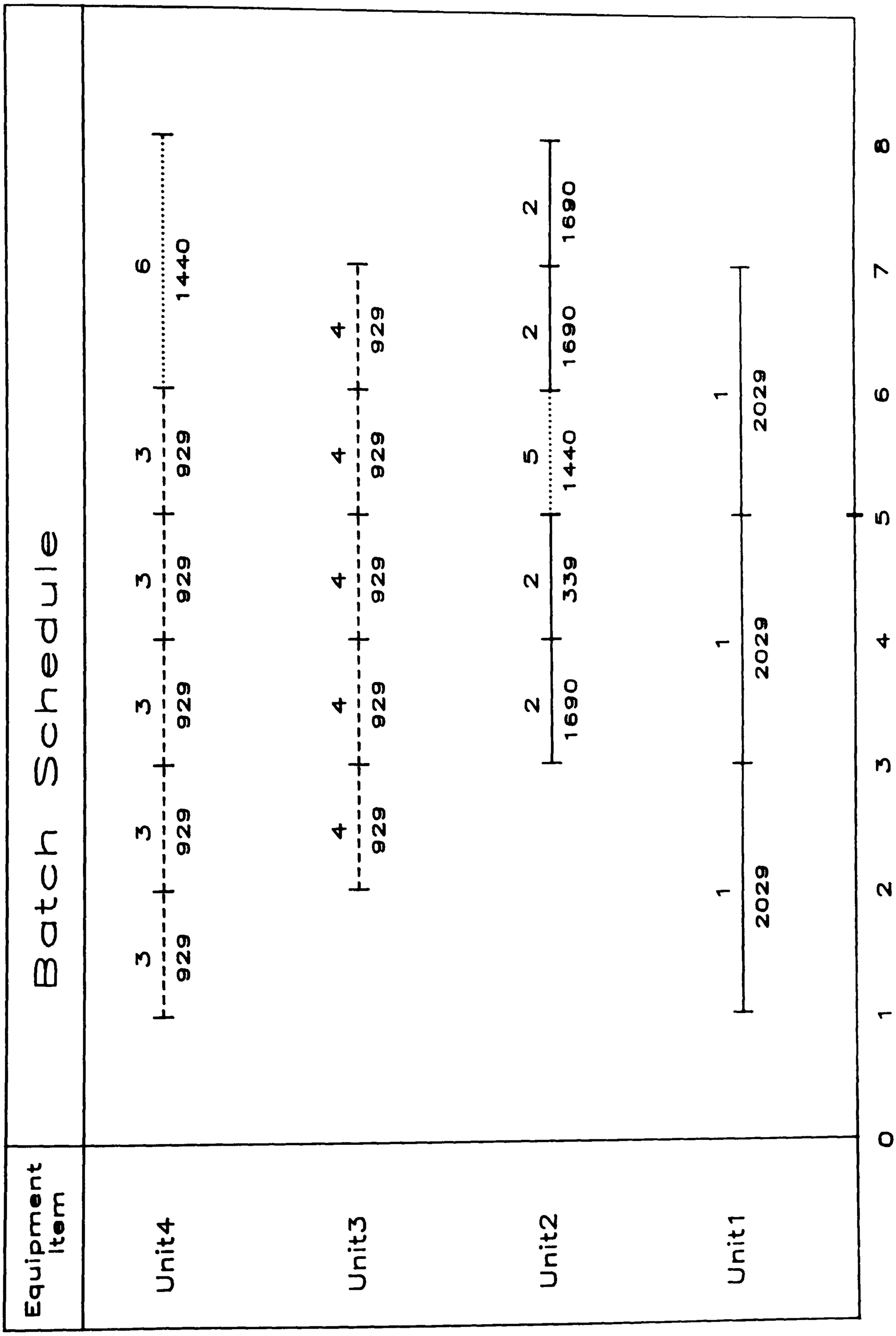


Figure 5.10 Optimal schedule for Problem 3

5.5 Batch System Problem 4

Four products are to be produced under the processing scheme shown in Figure 5.11. The specifications of the problem are given in Table 5.7.

Table 5.7 Specifications for Problem 4

Time Horizon = 8 hours			
Process Units	Size(kgr)	Units Suitability	Tasks processing time (hrs)
Unit 1	1000	Task 1	1 hour
Unit 2	2500	Task 3, Task 7	1 hour
Unit 3	3500	Task 4	1 hour
Unit 4	1500	Task 2	1 hour
Unit 5	1000	Task 6	1 hour
Unit 6	4000	Task 5, Task 8	1 hour
Process States		Storage Capacity (kgr)	Initial amount stored(kgr)
Feeds 1, 2, 3		unlimited	unlimited
Int 4		1000	0.0
Int 5		1000	0.0
Int 6		1500	0.0
Int 7		2000	0.0
Int 8		0	0.0
Int 9		3000	0.0
Product 1, 2, 3, 4.		unlimited	0.0
Objective: maximize total production $=(S_{P_1} + S_{P_2} + S_{P_3} + S_{P_4})_8$			

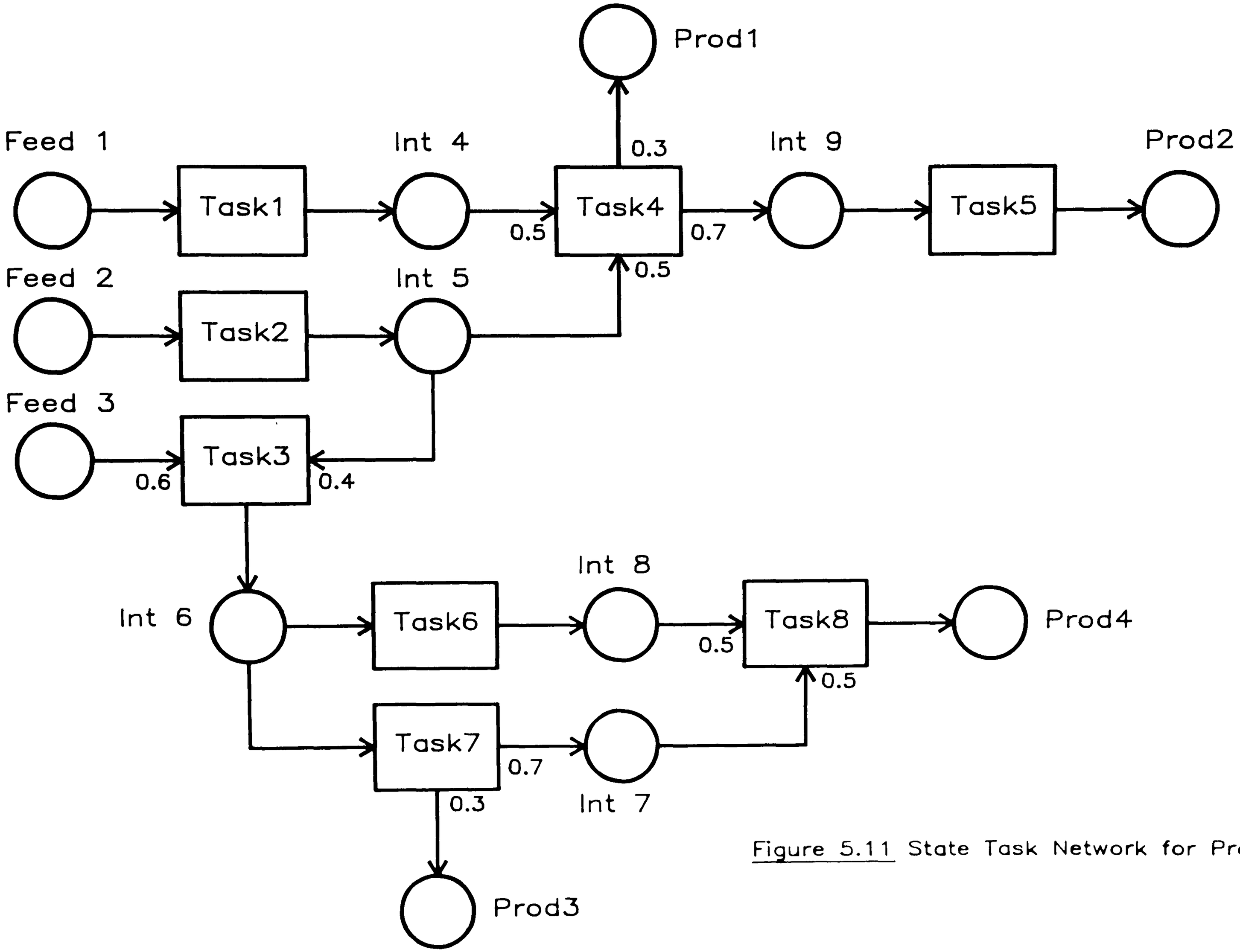


Figure 5.11 State Task Network for Problem 4

The initial solution generating algorithm of Section 4.3.1 has produced the schedule of Figure 5.12 and a corresponding production of 8875 kgr.

The optimal schedule of the process is shown in Figure 5.13. Plots of the inventory level vs time are given in Figure 5.14 for some of the states of the process.

Finally the dimensionality of the problem and the optimal production are given in Table 5.8.

Table 5.8 Dimensionality and Results for Problem 4

Number of integer vars.	64
Number of cont. vars.	181
Number of constraints	232
Optimal production (kgr)	14687.1
CPU sec on VAX 8600	3500

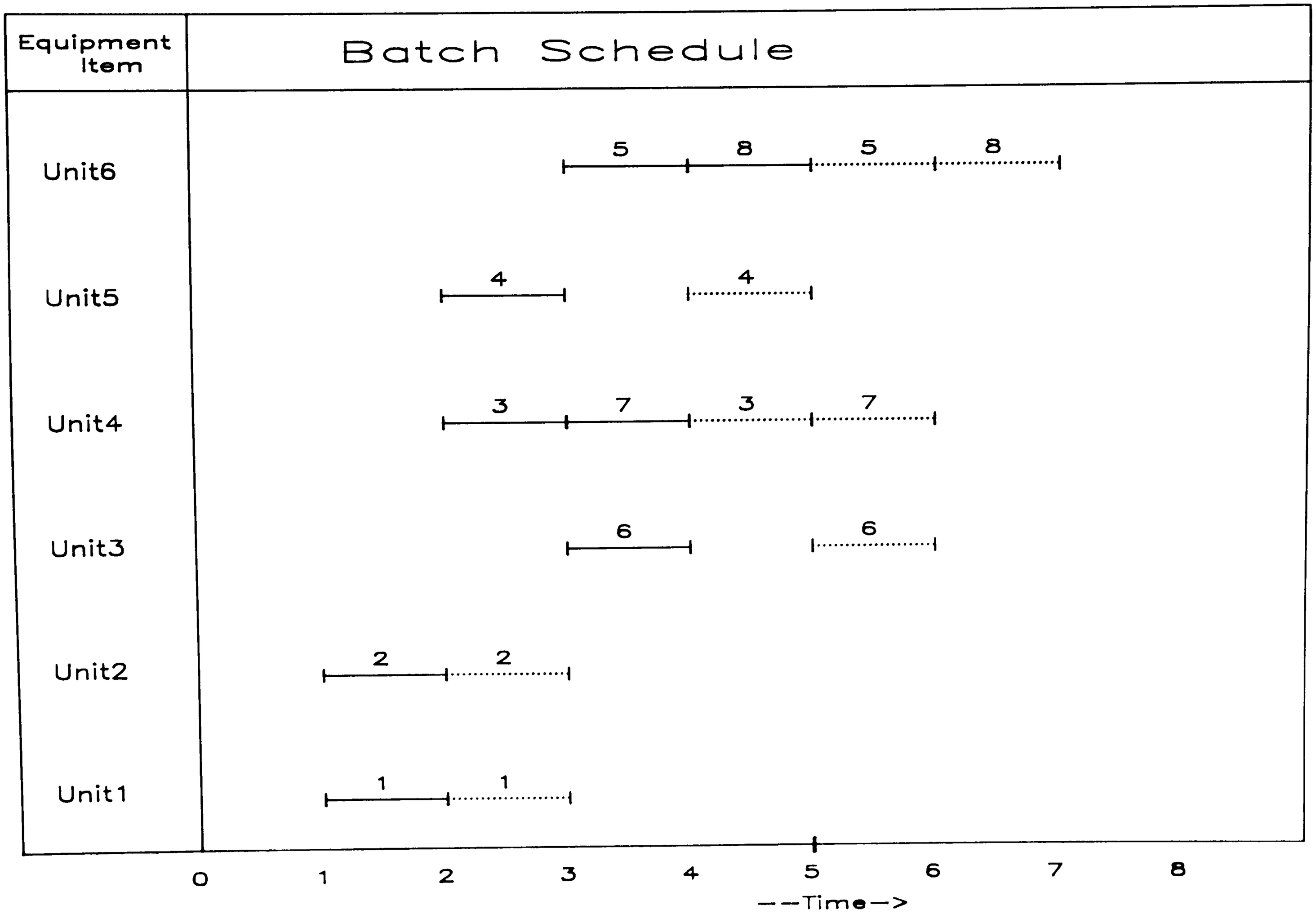


Figure 5.12 Initial solution for Problem 4

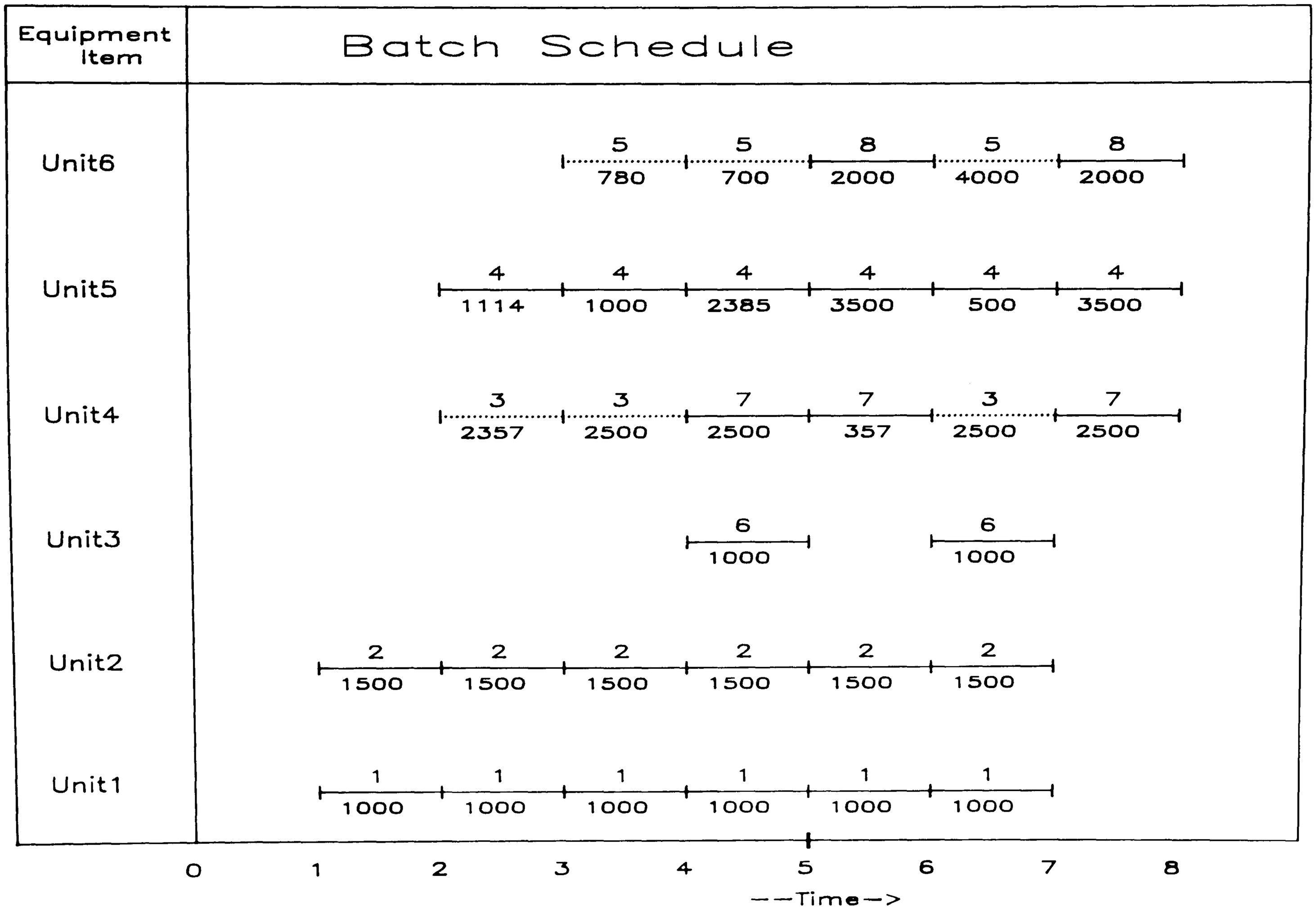
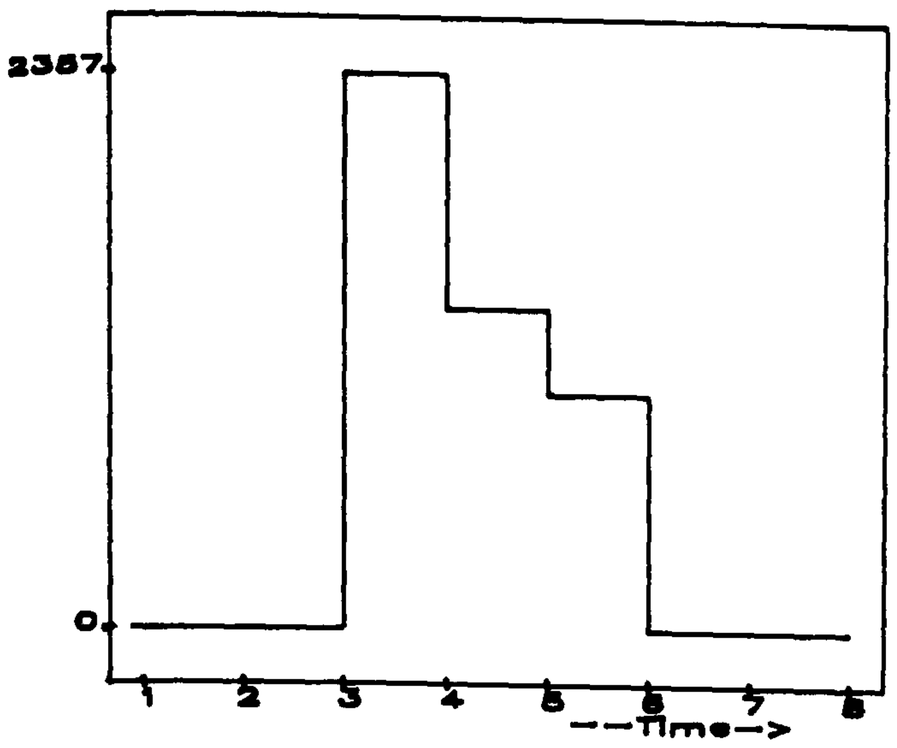
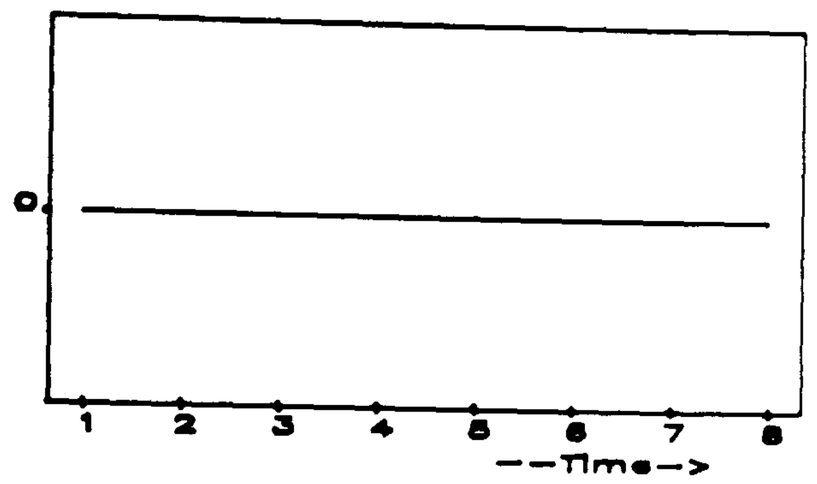
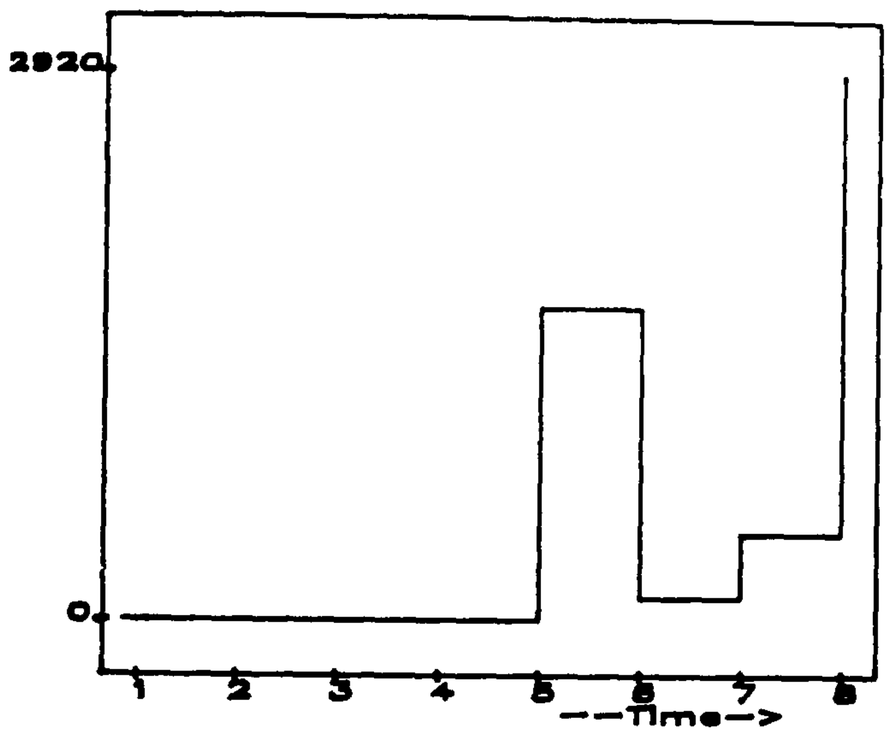
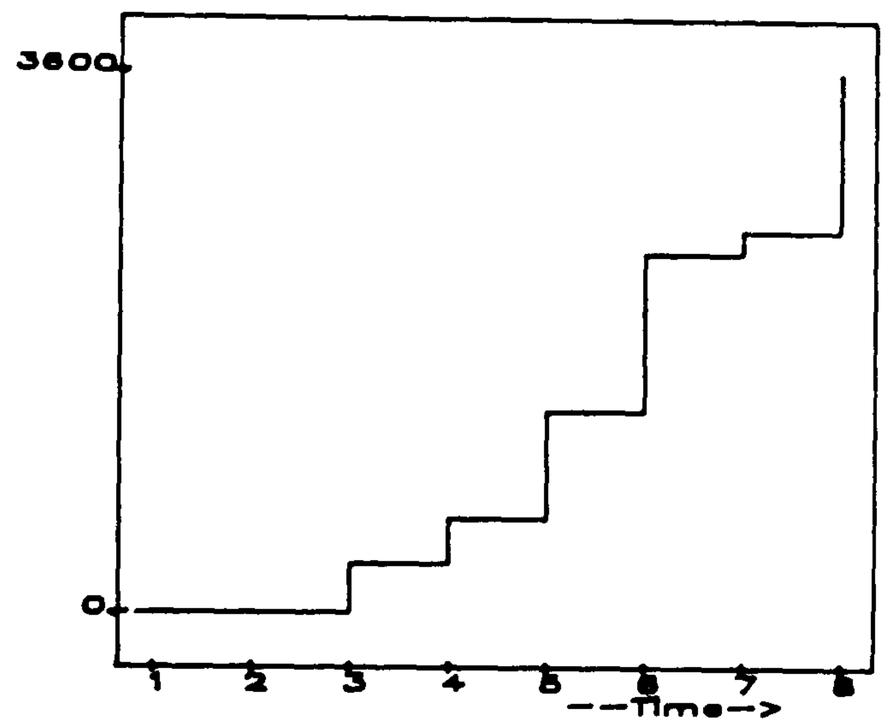


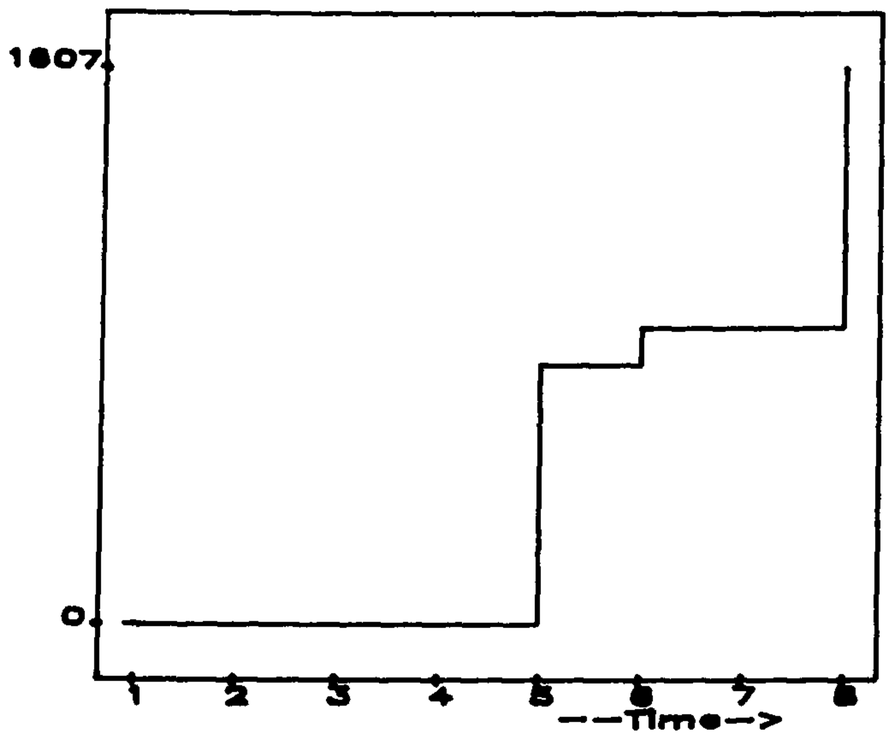
Figure 5.13 Optimal schedule for Problem 4











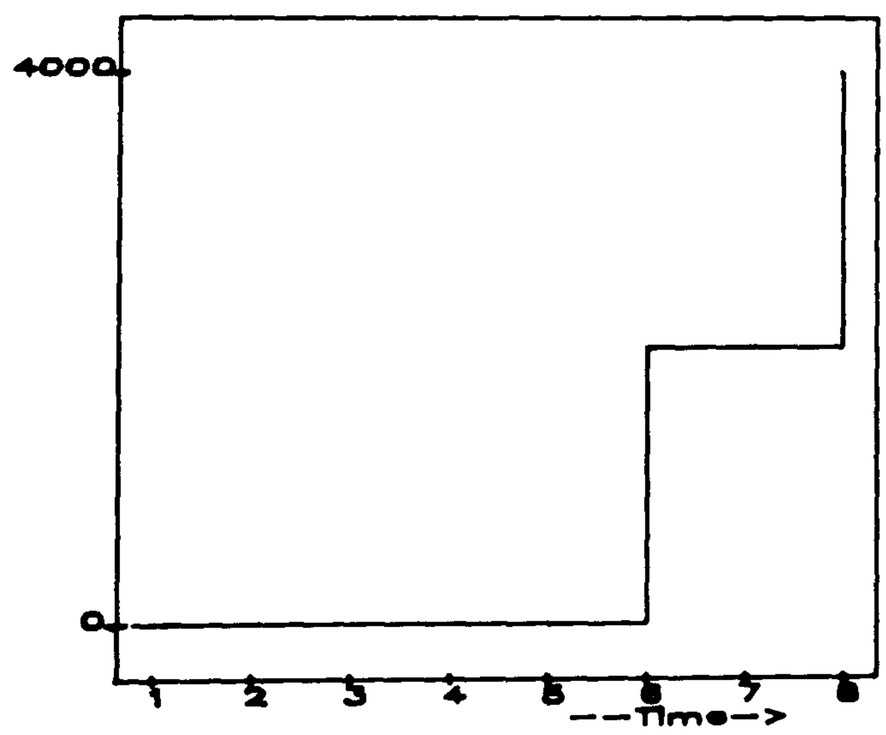


Figure 5.14 Inventory levels for Problem 4

5.6 Comparison of the algorithm with a commercial MILP package

A small size problem is considered in order to compare the behaviour of the branch and bound algorithm developed (version 2) with a general purpose MILP package (ZOOM).

The same problem has been solved by using two different LP libraries in version 1 of the algorithm, in order to investigate the dependence of the computing time on the LP method. The libraries are the XMP, incorporated into ZOOM package, and MINOS.

The dimensionality of the problem considered and the results of the above experiments are shown in Table 5.9.

Table 5.9 Comparison of branch and bound algorithm/ZOOM and LP libraries XMP/MINOS.

	Algorithm Version 2	ZOOM	Algorithm Version 1	
			XMP	MINOS
Number of integer vars.	36			
Number of cont. vars.	71			
Number of constraints	102			
Number of LPs solved	222		351	351
CPU sec on IBM 4341	62.3	993.7	262.54	152.23

The MILP package used (ZOOM) employs the branch and bound search to find the optimal solution and essentially follows the steps of the algorithm described in Section 4.2. It is a general use MILP package and, since it pays no regard to the characteristics of the problem, it takes much longer computing time than Algorithm BB2.

The comparison between XMP and MINOS linear programming libraries proves MINOS to be more efficient. This aspect is discussed in more detail in the next Chapter.

5.7 Discussion of the Results

From a number of problems examined conclusions can be drawn related to the behaviour of the physical system and the solution technique.

In previous research carried out in the field, the optimal allocation of the units is determined independently of the objectives imposed on the system. However, the criteria under which optimization is sought play a major role in the allocation of the units and the same problem may have completely different assignments of units to tasks for different objectives.

Another assumption very commonly made, that batches of material keep their identity (ie. are not split or mixed with other batches) restricts the alternatives of the process drastically and possibly excludes the optimal solution. The suggested formulation allows tasks to be performed (provided that there is material in their feed states), without it being necessary to follow their precedence constraints. From the results of the examples described it is shown that this may happen in the optimal schedule (eg. Problems 2, 2a and 4).

Parallel units are allocated to process the same tasks in Problems 2 and 2a. The amount of material which is processed by each unit is determined by the operating cost of the unit for this particular task.

The complexity of the mathematical problem is mainly affected by the number of potential alternatives among which the optimal solution is found. For example, the number of integer variables depends on the number of different tasks each of the units can perform. Moreover, availability of intermediate storage increases the flexibility of the plant and consequently the number of alternative choices. This explains the increased number of nodes which had to be explored in Problem 2 compared to the ones of Problem 2a.

However, the complexity of the MILP problem may also be affected by factors other than the number of integer variables. For example, Problem 4 has proved to be more complex to solve than Problem 2, in spite of the fact that the number of integer variables in Problem 4 is less than that in Problem 2. An explanation for the above, though not rigorous, could be the increased complexity of the State Task Network of Problem 4 compared to the one of Problem 2.

Although the algorithm prevents conflicting assignments (ie. a unit to be allocated to a task while it is processing another task at the same time), infeasibilities may be encountered in the course of the search due to violating constraints involving continuous variables (eg. storage capacity limitations). In this case the corresponding LP problem is infeasible and the branch is fathomed. Extending the argument, it should be mentioned that if the initial LP is infeasible, the MILP is infeasible too. This will happen if the problem is badly posed; therefore, the consistency of the specifications must be checked.

The second version of the algorithm is the most efficient one. It may seem surprising that the third version of the algorithm has poorer performance than the second. However, in the development of the algorithms the most important factor is the ability to evaluate various subproblems without solving them, by detecting the subassignment relationship. In the first step of the algorithm, where the various assignments are generated, the option of a unit remaining idle always follows the option of the unit being allocated to a task. Thus, in a depth-first search all the subproblems associated with sets of assignments $\sigma_i \subseteq \sigma_j$ are explored after the assignments σ_j have been explored and stored in the corresponding comparison-lists. Therefore the depth-first search favours the detection of the subassignment relationship and, consequently, the solution of the corresponding LPs is avoided. In the third version of the algorithm this order is destroyed and the advantage of exploring the most promising nodes first is not enough to compensate the fact that many of these subproblems would not need to be solved.

Comparisons of the developed algorithm with a general purpose MILP package reveals its superiority. This was expected since the algorithm is specially designed to exploit the structure of the system, whereas ZOOM is an MILP package for general use.

Finally, the importance of using an efficient LP method should be pointed out, since a large number of LP solutions are needed before reaching the optimum. The LP problems which are solved differ slightly from each other, mainly on the bounds of the variables. MINOS has the advantage of storing the basis of the optimal solution of a problem and using it when it tries to solve a similar one. Thus, it usually needs

only a few iterations to recover optimality in each node. Furthermore, when the nodes of the tree are explored, singularities may be encountered due to the redundancy of those constraints which have all the integer variables fixed. MINOS has been proved robust enough to deal with this situation.

C H A P T E R S I X

GENERAL CONCLUSIONS

The optimal production planning and scheduling problem of batch chemical processes has been considered in this thesis. The problem has been defined in such a way that a wide variety of process characteristics as well as targets which the batch chemical plant has to meet can be accommodated. Consequently, the proposed solution technique is applicable to a wide range of processes.

The representation of batch chemical processes with the State Task Network has been established. The explicit consideration of the status of material between the processing tasks (states) which is the key feature of the established representation, allows any level of complexity in the operating scheme, like common intermediate products, mixing, splitting and recycling of material within the process and tasks which have different time requirements for each of their outputs. Furthermore, since the intermediate states in a sense represent the in-process inventory, any combination of intermediate storage policies can be considered.

An important point to appreciate is that the State Task Network represents the physical process itself without making any explicit reference to where the operations are carried out. However, each processing task has associated with it a list of suitable equipment items. Solution of the production planning and scheduling problem involves deciding upon the allocation of equipment items to tasks throughout a prespecified time interval as well as the amounts of material which are processed and stored in the various stages of the process.

Having defined the problem to be solved and having established a suitable representation of the process, the next step has been the formulation of the corresponding mathematical model. The model is based on the discretization of time and the introduction of decision and continuous variables which sufficiently describe the process and result in linear constraints and objective function. More specifically, the decision (zero/one) variables indicate whether or not a process unit performs a particular task at a certain time period, whereas the continuous variables represent the various batch sizes and the inventory levels. The advantage of the introduction of this type of decision variables is that the problems of finding the optimal configuration of the process and optimizing with respect to the continuous variables are considered simultaneously; thus the need to iterate between them is circumvented.

The basic constraints of the model include resolution of conflicting demands on the units, capacity restrictions for the units and the intermediate storage stations and material balances in the states. However, other types of operational constraint can be incorporated in the model without adding any further complexity. As examples we could mention continuous/discrete utilities or raw material availability, due dates and precedence constraints for the products and fixed or sequence dependent cleaning and set-up times for the units. Essentially the incorporation of these constraints in the model tightens the feasible region and makes the search of the solution easier by reducing the number of potential alternatives. It is suggested that the incorporation of these constraints in the model should be implemented so that their effects on the complexity of the problem will be revealed.

The criteria under which the performance of a batch chemical plant has to be optimized vary from one plant to another and, moreover, may change with time within the same batch operating system. The formulated model can accommodate a variety of system's performance criteria. When the selected objective criterion is a cost function, all the significant costs can be taken into account, such as utilization and operating costs of the units, inventory and utility costs.

Another advantage of formulating the scheduling problem as an MILP is that it allows the study of the effects that various process or problem parameters have on the behaviour of the system by adding, deleting or modifying the constraints of the model.

The dimensionality of the MILP model depends on the structure of the State Task Network and the number and suitability of the available equipment items.

The algorithm developed for the rigorous solution of the MILP uses the branch and bound method and is specially designed to exploit the structure of the problem. Consideration of physical characteristics of the system resulted in the development of rules which allow a large number of nodes of the search tree to be implicitly enumerated, thus resulting in a substantial enhancement of the efficiency of the algorithm.

In commercial packages which employ the branch and bound method to solve MILP problems, heuristic rules are introduced to accelerate the search (Taha, 1975; Forrest,1974). In the algorithm developed in the present work, the combination of the dominance property of the system with a heuristic rule related to the branching order in the search tree did not prove to be effective. However, there may be other rules, either dictated by physical considerations of the system or by its mathematical structure,

the implementation of which could enhance the efficiency of the algorithm. Therefore, further research in this direction should be encouraged.

Furthermore, the idea of using strong cutting planes for solving mixed integer programming problems is gaining momentum and its effectiveness is starting to show (Van Roy, Wolsey, 1987). The use of cutting planes for solving our MILP could be effective, particularly in reducing the gap of the LP relaxation. Therefore, the study of the combination of strong cutting planes and our branch and bound strategy is encouraged.

As an exact solution method, the algorithm may be computationally intensive for large size problems. However, this is a general characteristic of all the techniques developed for solving MILP problems. In spite of the extensive theoretical research which has been carried out in this field during the past two decades, the progress in the computational aspects of integer programming has been slow and there is no known algorithm which shows uniformly efficient behaviour for all the problems. In this context it is believed that the branch and bound algorithm developed in the present work performs better than most general techniques, since it has been specially designed to take into account the structure of the problem.

In conclusion, the proposed strategy provides a versatile tool for the systematic treatment of scheduling problems arising in batch chemical processes with a wide variety of potential features and/or operating targets.

REFERENCES

1. Baker, K.R., "Introduction to Sequencing and Scheduling", J. Wiley, USA, (1974)

2. Baker, K.R., "A Branch and Bound Network Algorithm for Interactive Process Scheduling", Math. Pr. Studies 15, (1981).

3. Bakshi, M.S., "The Sequencing Problem", Man. Sci., 16, No. 4 (1969).

4. Bowman, E.H., "The Schedule Sequencing Problem", Oper. Research, 7, No. 5 (1959).

5. Bradley, S.P., Hax, A.C., Magnanti, T.L., "Apllied Mathematical Programming", Addison-Wesley, USA, (1977).

6. Conway, R.H., Maxwell, W.L., Miller, L.W., "Theory of Scheduling", Addison-Wesley, USA, (1967).

7. Cristophides, N., "Graph Theory: An Algorithmic Approach", Academic Press, New York, (1975).

8. Egli, U.M., Rippin, D.W.T., "Short Term Scheduling for Multiproduct Batch Chemical Plants", Comp. Chem. Eng., 10, No. 4, (1986).

9. Forrest, J.J.H., Hirst, J.P.H., Tomlin, J.A., "Practical Solution of Large MILP Problems with UMPIRE", Man. Sci., 20, No. 5, (1974).

10. Garfinkel, R.S., Nemhauser, G.L., "Integer Programming", John Wiley, USA, (1972).

11. Giffler, B., Thompson, G.L., "Algorithms for Solving Production Scheduling Problems", Oper. Res. 8, No. 4, (1960).

12. Graves, S.C., "A Review of Production Scheduling", Oper. Res 29, No. 2, (1981).

13. Greenberg, H.H., "A Branch and Bound Solution to the General Scheduling Problem", Oper. Res 16, No. 2, (1968).

14. Grossmann, I.E., Sargent, R.W.H., "Optimum Design of Multipurpose Chemical Plants", Ind. Eng. Chem. Pr. Des. Dev., 18, No. 2, (1979).

15. Imai, M, Nishida, N., "New Procedure Generating Suboptimal Configurations to the Optimal Design of Multipurpose Batch Plants", Ind. Eng. Chem. Pr. Des. Dev., 23, No. 4 (1984).

16. Janicke, W., "On the Solution of Scheduling Problems for Multipurpose Batch Chemical Plants", Comp. Chem. Eng., 8, No. 6, (1984).

17. Janicke, W., "Optimal Assignment of Products to Equipment Items in Multipurpose Batch Chemical Plants", Comp. Chem. Eng., 11, No. 3, (1987).

18. Karimi, I.A., Reklaitis, G.V., "Intermediate Storage in Noncontinuous Processes Involving Stages of Parallel Units", AIChE J. 31, No. 1, (1985).

19. Klossner, J., Rippin, D.W.T., "Combinatorial Problems in the Design of Multiproduct Batch Plants: Extension to Multiplant and Partly Parallel Operations", A.I.Ch.E. Annual Meeting, San Francisco, (Nov. 1984).

20. Knopf, F.C., "Sequencing a Generalised Two Stage Flowshop with Finite Intermediate Storage", Comp. Chem. Eng., 9, No. 3, (1985).

21. Knopf, F.C., Okos, M.R., Reklaitis, G.V., "Optimal Design Of Batch/Semicontinuous Processes", Ind. Eng. Chem. Pr. Des. Dev., 21, No. 1, (1982).

22. Ku, H.M., Karimi, J.A., "Scheduling in Multistage Serial Batch Processes with Finite Intermediate Storage. Parts I, II", Paper No. 72e, A.I.Ch.E. Annual Meeting, Miami, (Nov. 1986).

23. Lazaro, M. Puigjaner, L., "Simulation and Optimization of Multiproduct Plants for Batch and Semibatch Operation", J. Chem. Eng. Symp. Ser. 92, (PSE 85).

24. Lawler, E.L., Wood, D.E., "Branch and Bound Methods. A Survey", Oper. Res 14, No. 4 (1966).

25. Loonkar, R.Y., Robinson, J.D., "Minimization of Capital Investment for Batch Processes", Ind. Eng. Chem. Pr. Des. Dev., 9, No. 4, (1970).

26. Manne, A.S., "On the Job-Shop Scheduling Problem", Oper. Res 8, No. 2, (1960).

27. Marsten, R.E., "The Design of the XMP Linear Programming Library", ACM Trans. on Math Software, 7, No. 4, (1981).

28. Mauderli, A., Rippin, D.W.T., "Production Planning and Scheduling for Multipurpose Batch Chemical Plants", Comp. Chem. Eng., 3, (1979).

29. Mauderli, A., Rippin, D.W.T., "Sceduling Production in Multipurpose Batch Plants: The BATCHMAN Program", CEP, (April 1980).

30. Murtagh, B.A., Saunders, M.A., "MINOS 5.1 User's Guide", Technical Report.

31. Pritsker, A.A.B., Watters, L.J., Wolfe, P.M., "Multiproject Scheduling with Limited resources. A Zero-One Programming Approach", Man. Sci., 16, No. 1, (1969).

32. Reklaitis, G.V., "Review of Scheduling of Batch Operations", A.I.Ch.E. Symp. Ser., No. 214, Vol. 78.

33. Rich, S.H., Prokopakis, G.J., "Scheduling and Sequencing of Batch Operations in a Multipurpose Plant", Ind. Eng. Chem. Pr. Des. Dev. 25, No. 4, (1986).

34. Rippin, D.W.T., "Computer Aided Design and Scheduling for Batch Processes. A Review of Current Status and Future Prospects", Ins. Chem. Eng. Symp. Ser. No. 94, (1985).

35. Rippin, D.W.T., "Design and Operation of Multiproduct and Multipurpose Batch Chemical Plants. An Analysis of the Problem Structure", Comp. Chem. Eng., 7, No. 4, (1983).

36. Rippin, D.W.T., "Problems in the Design and Operational Planning for Multiproduct Batch Plants", presented in Austria, (July 1985).

37. Rippin, D.W.T., "Review Paper: Simulation of Single and Multiproduct Batch Chemical Plants for Optimal Design and Operation", Comp. Chem. Eng. 7, No. 3, (1983).

38. Robinson, J.D. Loonkar, R.Y., "Minimizing Capital Investment for Multiproduct Batch Plants", Proc. Tech. Int., 17, No. 11, (1972).

39. Sisson, R.L., "Methods of Sequencing in Job-Shop. A review", Oper. Res 7, (1959).

40. Sparrow, R.E., Rippin, D.W.T., Forder, G.J., "MULTI-BATCH: A Computer Package for the Design of Multiproduct Batch Plants", The Chemical Engineer, 289, (Sept. 1974).

41. Sparrow, R.E., Forder, G.J., Rippin, D.W.T., "The Choice of Equipment Sizes for Multiproduct Batch Plants. Heuristics vs. Branch and Bound", Ind. Eng. Chem. Pr. Des. Dev., 14, No. 3, (1975).

42. Suhami, I., Mah, R.S.H., "An Implicit Enumeration Scheme for the Flowshop Problem with No Intermediate Storage", Comp. Chem. Eng., 5, (1981).

43. Suhami, I., Mah, R.S.H., "Optimal Design of Multipurpose Batch Plants", Ind. Eng. Chem. Pr. Des. Dev., 21, (1982).

44. Taha, H.A., "Integer Programming. Theory, Application and Computations", Academic Press (1975).

45. Taha, H.A., "Operations Research. An Introduction", Collier, Macmillian, USA (1982).

46. Takamatsu, T., Hashimoto, I., Hasebe, S., "Optimal Design and Operation of a Batch Process with Intermediate Storage Tanks", Ind. Eng. Chem. Proc. Des. Dev. 21, (1982).

47. Van Roy, T.J., Wolsey, L.A., "Solving Mixed Integer Programming Problems Using Automatic Reformulation", Oper.Res 35, No. 1, (1987).

48. Vaselenak, J.A., Grossmann, I.E., Westerberg, A.W., "An Embedding Formulation for the Optimal Scheduling and Design of Multipurpose Batch Plants", Ind. Eng. Chem. Res., 26, No. 1, (1987).

49. Vaselenak, J.A., Grossmann, I.E., Westerberg, A.W., "Optimal Retrofit Design of Multiproduct Batch Plants", Ind. Eng. Chem. Res., 26, No. 4, (1987).

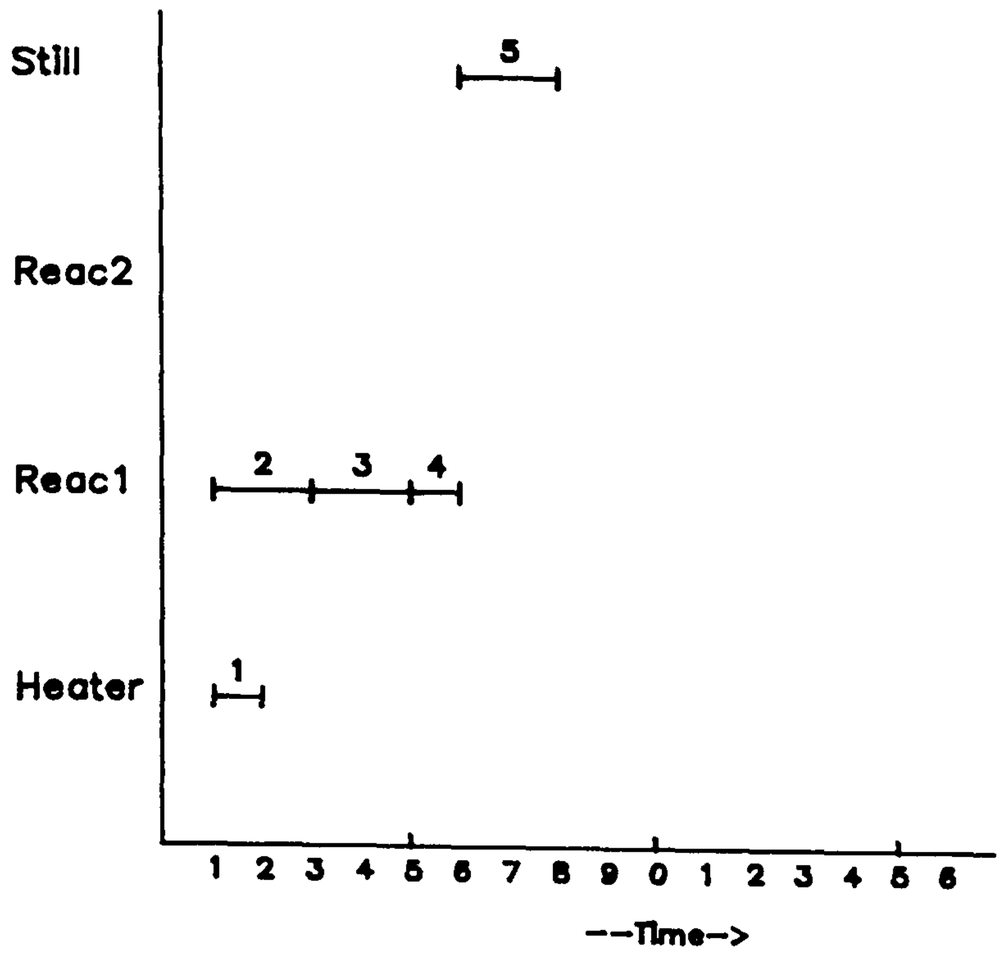
50. Wagner, H.M., "An Integer Linerar Programming Model for Machine Scheduling", Nav. Res. Log., Quat., 6, No. 2, (1959).

APPENDIX A

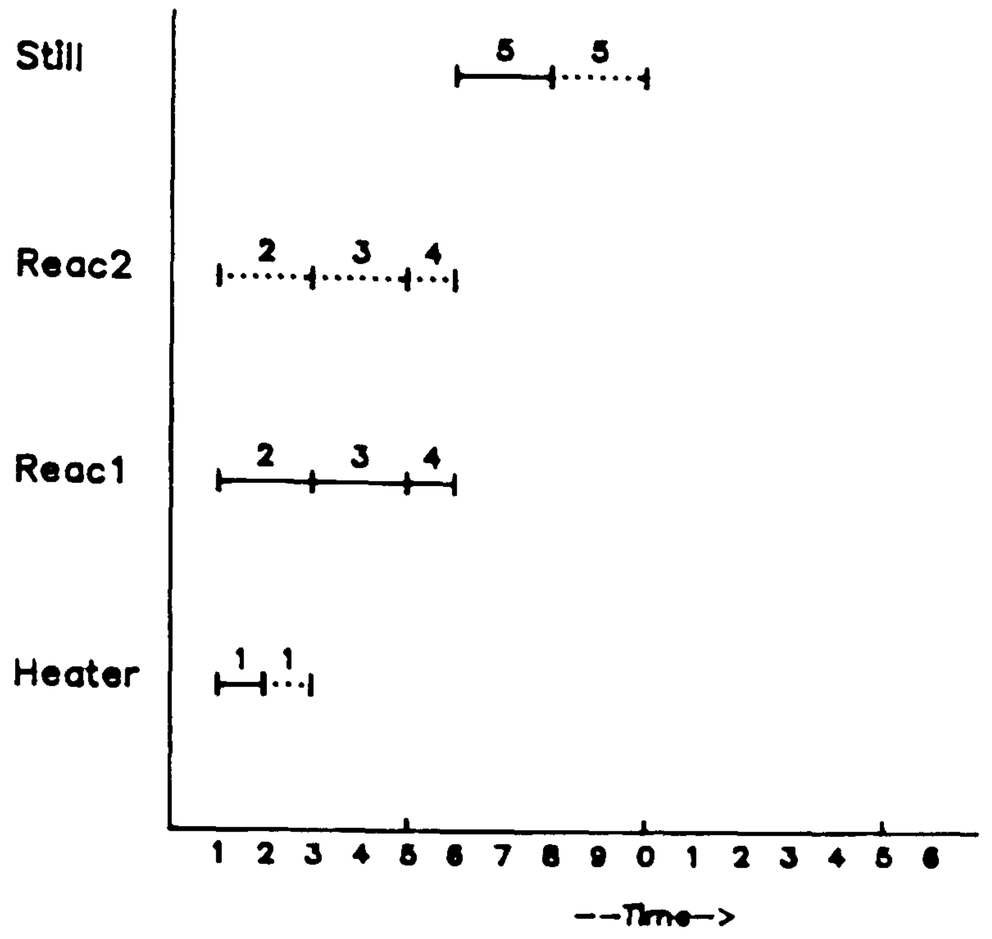
Example for the initial solution generating algorithm

As mentioned in Section 4.3.1 the initial solution generating algorithm (ie. Algorithm A0) considers the process to be carried out as a number of passes of material (batch sequences) through the State Task Network. Each batch sequence is scheduled independently of all the subsequent ones. However, the schedule of all the batch sequences except the first one is affected by the schedules of the preceding sequences, since these determine the time intervals during which the process units are available.

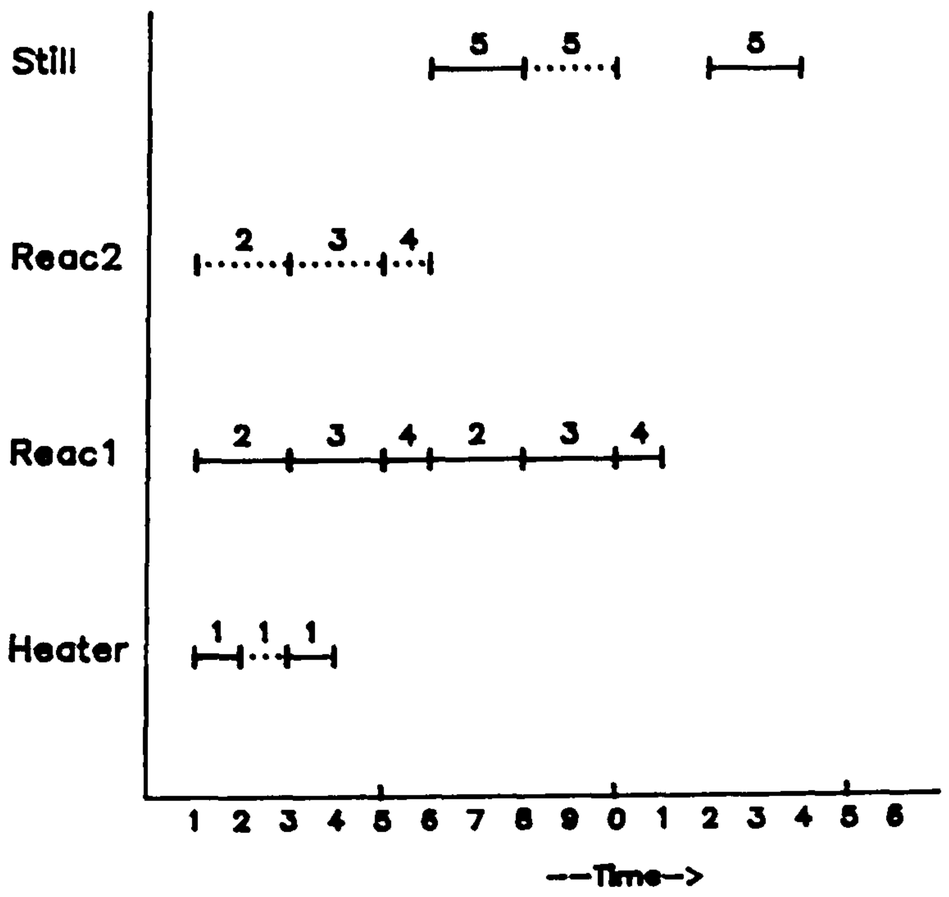
An example illustrating how the algorithm proceeds is shown in Figure A.1. The batch process of Problem 2 of the fifth Chapter is considered. Figure A.1a shows the schedule of the first batch sequence. As can be seen, Tasks 1 and 2 are performed simultaneously in the Heater and Reactor 1 respectively. After they have both been completed, Tasks 3, 4 and 5 are carried out in turn. In Figure A.1b a second batch sequence is scheduled (dotted lines). Tasks 3, 4 and 5 of the second sequence are performed in Reactor 2, since Reactor 1 is occupied. Finally, Figures A.1c and A.1d show the units allocation over time when a third and a fourth batch sequence are scheduled.



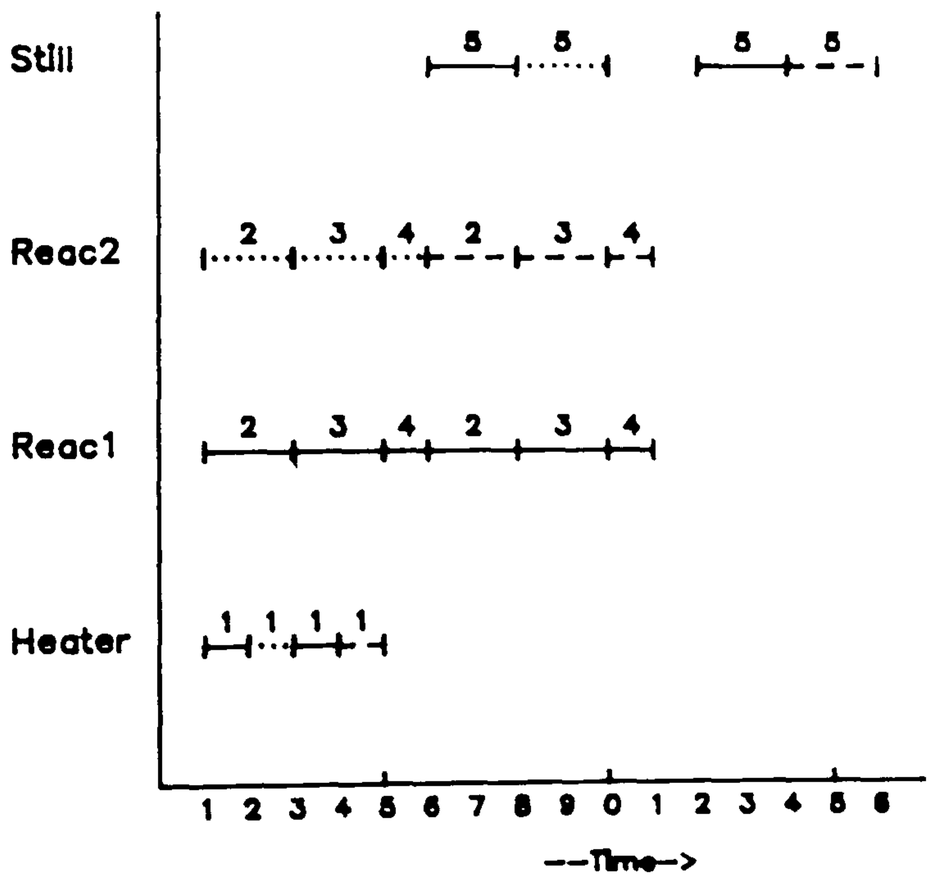
a.



b.



c.



d.

Figure A.1 Example for the generation of initial solution

APPENDIX B

Implementation Aspects

A PASCAL program has been developed which consists of a set of procedures for the input data control, the scheduling algorithm and the generation of the solution reports. The program is linked to the MINOS large scale linear programming code. The input data for the various LP problems created during the branch and bound search are generated by a procedure and transferred to MINOS internally.

In order to facilitate the construction of the input file by the user, the required data are supplied in the form of a specially designed language corresponding closely to the State Task Network. More specifically, the input file of the program comprises the time horizon specification and the description of the processing tasks, states and equipment items in turn. As an example, the input file for Problem 2 of the fifth Chapter is given in the following pages. A comprehensive set of consistency checks are performed on the input data before the scheduling algorithm itself is activated.

Input File for Scheduling Problem 2

```
* Time horizon definition

TIME  10

* Processing Task Definition
* Instate S x      :  Input from state S; fraction of total input x.
* Outstate S x y   :  Output into state S; fraction of total output x;
*                      processing time y.
* Equipment x      :  Equipment item suitable for this task; equipment
*                      operating cost.
TASK Heat
     INSTATE    FeedA   1.0
     OUTSTATE   HotA    1.0       1
     EQUIPMENT  Heater  0.2
TASK Reaction 1
     INSTATE    FeedB   0.5
     INSTATE    FeedC   0.5
     OUTSTATE   IntBC   1.0       2
     EQUIPMENT  Reac1   0.32
     EQUIPMENT  Reac2   0.5
TASK Reaction 2
     INSTATE    HotA    0.4
     INSTATE    IntBC   0.6
     OUTSTATE   IntAB   0.6       2
     OUTSTATE   Prod1   0.4       2
     EQUIPMENT  Reac1   0.7
     EQUIPMENT  Reac2   0.5
```

```
TASK Reaction 3

     INSTATE     FeedC    0.2

     INSTATE     IntAB    0.8

     OUTSTATE    IntE     1.0       1

     EQUIPMENT   Reac1    0.2

     EQUIPMENT   Reac2    0.3

TASK Separation

     INSTATE     IntE     1.0

     OUTSTATE    Prod2    0.9       1

     OUTSTATE    IntAB    0.1       2

     EQUIPMENT   Still    0.0


* Process States Definition

* Price     :  Value attached to unit amount of material in this state at
*              final time;

* Initial   :  Amount initially available in this state;

* Capacity  :  Storage capacity available for material in this state.

STATE FeedA

     PRICE       0.0

     INITIAL     100000

     CAPACITY    1000000

STATE FeedB

     PRICE       0.0

     INITIAL     100000

     CAPACITY    100000

STATE FeedC

     PRICE       0.0

     INITIAL     100000
```

```
     CAPACITY    100000
STATE HotA
     PRICE       0.0
     CAPACITY    100
STATE IntAB
     PRICE       0.0
     CAPACITY    200
STATE IntBC
     PRICE       0.0
     CAPACITY    150
STATE IntE
     PRICE       0.0
     CAPACITY    200
STATE Prod1
     PRICE       10.0
     CAPACITY    100000
STATE Prod2
     PRICE       10.0
     CAPACITY    100000

* Equipment Item Definitions

* Capacity x y :  Equipment capacity; equipment usage cost.

UNIT Heater
     CAPACITY    100.0    0.0
UNIT Reac1
     CAPACITY    50.0     0.0
```

```
UNIT Reac2

     CAPACITY    80.0     0.0

UNIT Still

     CAPACITY    200.0    0.0


* End of Data
```