

**SHORT TERM PRODUCTION SCHEDULING
FOR A TWO-STAGE MANUFACTURING SYSTEM**

by

Khaled BELARBI

A thesis submitted to the
the Victoria University of Manchester
in fulfilment of the requirement for the degree of
Doctor of philosophy
in the Faculty of Technology

DECEMBER 1987

**Control Systems Center
University of Manchester
Institute of Science and Technology**

Abstract

This thesis deals with the lower level of decision in a hierarchical approach to the planning and scheduling of production for two-stage manufacturing facilities. It is concerned with the short term scheduling of the item lot sizes output from the higher level. The plants concerned consist of two stages of activities: at the first stage, the products are manufactured, at the second stage, the manufactured products are packed in different formats. Between the two stages, there are intermediate storage facilities. In the approach adopted, the problem is decomposed into two subproblems, solved sequentially: the first is concerned with the scheduling of the lot sizes on the packing lines, the second is concerned with the scheduling of the manufacturing units and intermediate storage and has as input the packing lines schedule.

At the packing lines level, the lot sizes are to be loaded onto the packing lines so that changeover and packing costs are minimised. Due to connectivity constraints and shared manpower resources, the packing lines are interdependent. The problem is formulated as a pure integer problem and a branch and bound algorithm is proposed. Lower bounds are computed from a relaxation that decouples the problem into two subproblems: a machine loading subproblem, formulated as a general assignment problem and a pure sequencing subproblem formulated as the problem of finding the shortest arborescence through a certain graph. In order to improve the bound obtained, penalties are computed.

At the manufacturing units and intermediate storage level, the demand arising from the packing lines schedule is to be satisfied while minimising production and set-up costs. A tree search procedure that uses a Lagrangean relaxation of the original problem for computing lower bounds is proposed.

There is no guarantee that there will be a feasible manufacturing units schedule that satisfies the demand arising from the first optimal packing lines schedule. Thus a one pass procedure is not feasible. In this respect, a coordination device is introduced that allows the user to obtain overall feasibility by reconsidering one or the other schedule. The method can be seen as a multi-pass procedure whose overall target is feasibility rather than optimality.

Computation results are first reported for the packing lines algorithm only and then for the overall algorithm.

ACKNOWLEDGEMENTS

I would like to express my thanks to Professor Madan G. Singh for giving me the opportunity to work in this interesting field.

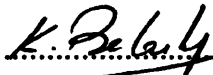
I am grateful to Dr. K.S. Hindi for his helpful suggestions and comments and for proof reading patiently the final draft.

I would like to thank all my friends at the control systems center and elsewhere for their support and kindness.

Finally, I am indebted to the Algerian Ministry of Higher Education for their financial support, without which this would not have been possible.

DECLARATION

No portion of the work in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or institution of learning.


K. BELARBI
DECEMBER 1987

A ma famille

After obtaining an 'ingenieur d'etat' degree in telecommunications at the University of Algiers, the author worked for four years as an assistant lecturer at the University of Constantine (Algeria). He then came to U.M.I.S.T where he first prepared an Msc in Control System (October 1983- October 1984) and started his Phd in November 1984.

Contents

1	Introduction	1
2	Integer optimisation	6
2.1	Introduction	6
2.2	Description of integer optimisation problems	7
2.2.1	Categorisation of integer problems	7
2.2.2	Formulation of integer optimisation problems	8
2.3	Solution methods for integer optimisation	14
2.3.1	Cutting planes methods	15
2.3.2	Tree search methods	15
2.3.3	Dynamic programming	16
2.3.4	Heuristics	18
2.4	Tree search procedures	18
2.4.1	Subproblem separation	19
2.4.2	Subproblems fathoming	20
2.4.3	Subproblem selection	21
2.5	Lagrangian relaxation	22
2.5.1	The basic results	23
2.5.2	Computing the multipliers	25
2.5.3	Lagrangian relaxation in tree search	27
3	Review of the relevant literature	29
3.1	Introduction	29

3.2	The production scheduling problem	30
3.3	Minimisation of a performance criterion	31
3.3.1	The single machine problem	33
3.3.2	Parallel machines	34
3.3.3	The flow shop and job shop problems	36
3.4	Problems with cost function	37
3.4.1	One product one facility	38
3.4.2	Multi-product single facility	42
3.4.3	Multi-product multi-facility	46
3.4.4	Multistage models	49
3.5	Summary	51
4	Problem description and solution methodology	53
4.1	Introduction	53
4.2	Description of the plants	54
4.2.1	General features	54
4.2.2	Manufacturing units	54
4.2.3	Intermediate storage	56
4.2.4	Material handling systems	56
4.2.5	Packing lines	57
4.2.6	Planning cycle and demand	58
4.3	The short term commitment	58
4.4	Solution methodology	60
4.4.1	Overall approach	60
4.4.2	The coordination method	61
5	Short term commitment for the packing lines	64
5.1	Introduction	64
5.2	Description of the system	65
5.3	A unrestricted problem formulation	67
5.3.1	A problem transformation	73

5.4	Relaxation and lower bounds for the transformed problem	74
5.4.1	Relaxation	74
5.4.2	The machine loading subproblem	76
5.4.3	The sequencing subproblem	82
5.4.4	Computing penalties on LB	84
5.5	The overall algorithm for the packing lines	86
5.5.1	Introduction	86
5.5.2	Branching rule	87
5.5.3	A heuristic solution	89
5.5.4	An outline of the algorithm	90
5.5.5	Handling the resource constraints	93
5.6	Conclusion	95
6	Short term scheduling for the production-intermediate storage	
	subsystem	97
6.1	Introduction	97
6.2	Description of the problem	98
6.2.1	Description of the systems	98
6.2.2	The short term commitment	99
6.3	The basic model	101
6.4	Dedicated storage	104
6.4.1	The Lagrangean relaxation	105
6.4.2	The one product subproblem	105
6.5	The solution procedure	112
6.5.1	Overview	112
6.5.2	Branching rule	112
6.5.3	The first node	113
6.5.4	General node	114
6.5.5	Backtracking	115
6.5.6	Updating parameters of the coordination process	115
6.6	A case with flexible storage	116

6.6.1	Extension of the basic model	116
6.6.2	The Lagrangean relaxation	117
6.6.3	The solution procedure	118
6.6.4	Branching rule	120
6.7	Conclusion	120
7	Results	122
7.1	Introduction	122
7.2	Packing lines algorithm	122
7.2.1	Problem (P5)	123
7.2.2	Problem (P6)	124
7.2.3	Problems (<i>P71</i>) and (<i>P72</i>)	124
7.3	Computational results for two-stage systems	125
7.3.1	Computational results for <i>PSP1</i>	128
7.3.2	Computational results for <i>PSP2</i>	131
8	Conclusion	161
	References	166
A	The knapsack problem	173
B	Algorithm for the shortest arborescence	177
C	Data for packing lines problems	179
D	Data for two-stage problems	186

Chapter 1

Introduction

This thesis is concerned with the short term scheduling of production in a two stage manufacturing facility. It is part of a larger study whose aim was to examine possible improvements of the performance of the manufacturing facilities of a major producer of detergents.

The primary goal was to develop computerised approaches that the company could use for its production planning and scheduling. These approaches must be capable of providing good solutions which take advantage of both the theoretical advances in the field and the advance in computers technology. Indeed, the management of the company believes that considerable savings could be achieved in this way.

Typically, the aim of the managers of any company that produces marketable goods is to try and satisfy the demand of customers at minimum operational costs, thus minimising waste, increasing productivity and bringing profits. The decisions to be taken in order to achieve this, will very often involve choosing among many alternatives.

Clearly, for a company manufacturing many products, the first question that arises is how much to manufacture of each product in the next planning horizon in order to satisfy the external demand, given the capacity of the factories in labour, machines and so on. The planning horizon is generally one year or one

season.

A simple policy is to produce exactly as required by the customers in each period. Although this policy is being applied in some cases, it involves very frequent changeovers of the production facilities and does not allow for safety storage. Moreover, in many cases, due to promotional sales and seasonal variations, the demand for some products can be very high in some periods and may substantially exceed the capacity of a given factory. Thus, it becomes necessary to produce more than needed in periods of low demand in order to build stocks which can be used in periods of high demand. However, holding products in inventory is often costly and can be very expensive in the case of perishable goods.

It is, therefore, desirable to find a policy which gives an acceptable compromise between set-up costs and inventory holding costs.

On the other hand, for a company using a number of facilities in a factory, it is necessary to decide on the exact moment in time to start and finish production for every product and what resources are allocated to its production and storage. In other words, it is necessary to establish a detailed schedule of production. Commonly, the best order in which the products will be manufactured is sought.

For all the above decisions to be carried out, the management must have a good forecast of the demand of the different customers for the various products and a provision of raw materials which avoids shortage and waste.

Supposing that a good forecast of the demand, which in itself is a difficult problem, is at hand and that the purchase of raw materials has been decided upon, the problem for the management is to plan and schedule production for the next horizon. It is well known [52,11] that the overall production planning and scheduling problem can be formulated in a linear programming framework. However, due to the size of the resulting problem, such a formulation is very often undesirable. Indeed, for the problem considered in this work, where decisions are sometimes taken at the hourly level, for a planning horizon of six to ten weeks, a monolithic model would involve a considerable number of variables

and constraints.

An alternative approach [35] is to exploit the natural hierarchy in the structure of the problem. In this approach, the problem is separated into a number of decision levels, such that the lower the level the more detailed it is and with each level providing data to and imposing constraints on the level below it. The overall problem is then tackled by solving sequentially the problems arising at each level.

The number of decision levels and the organisation of the hierarchy depend on the problem at hand. However, generally, a production planning and scheduling problem is separated into two problems: the aggregate planning problem and the detailed scheduling problem.

The aggregate planning problem is based on aggregate data and longer time periods. It deals with the determination of the quantity to be produced of each product in each time period, that is the lot sizes, so as to prevent underload or overload of the resources and satisfy the demand. The items may be aggregated into families and machines into machine centers. The decisions are taken over the long term, commonly one year or one season and the basic time period is, generally, one month or one week, depending on the horizon.

At the detailed scheduling level, given the lot sizes, the actual starting and finishing times of the production of each product and the resources allocated to produce it are determined for each time period. The decisions are taken over the short term, usually one month or one week, and the basic time period can be one week or one shift.

This thesis is concerned with the second level of this hierarchical approach, that is the detailed scheduling of production under resource constraints.

Although there have been many theoretical advances in the production scheduling field, manual scheduling practices are still predominant. In these cases, schedule evaluation depends on many criteria although flexibility and search for feasibility are the dominant features and optimality or near optimality are rarely obtained.

The work in this thesis is a contribution towards a more computerised approach to the detailed production scheduling problem. The emphasis will be on the use of operations research methods, in particular integer optimisation. The aim is to develop algorithms that generate good, feasible schedules of production which could be used in a class of plants.

The plants considered consist of two stages of activities. At the first stage, the products are manufactured, at the second stage the manufactured products are packed in different formats. Between the two stages, there are intermediate storage facilities.

In the approach adopted here, the problem is split into two subproblems: the first is concerned with the scheduling of the packing lines while the second is concerned with the scheduling of the manufacturing units and intermediate storage and has as input the schedule of the packing lines. However, there is no guarantee that there will be a manufacturing units schedule that satisfies the demand arising at the packing lines. This is mainly due to the fact that there are, often, a number of products that share the same manufacturing and storage facilities and to the difference in the operating speeds of the packing lines and the manufacturing units. A coordinating device is introduced which allows the user to obtain overall feasibility, by reconsidering one or the other schedule. The method can be seen as multi-pass procedure whose overall target is feasibility rather than optimality. The algorithms developed for each subproblem use integer optimisation techniques.

The thesis is organised as follows:

Since integer optimisation methods are used extensively in this thesis, the second chapter presents an overview of integer optimisation methods, with emphasis on branch and bound methods and Lagrangean relaxation as used in integer optimisation. Some well known results are presented for the latter.

In the third chapter a survey of the scheduling literature is presented. Because of the vast body of publications, this survey is restricted to deterministic problems, most relevant to the purpose of this thesis. The survey is divided into

two parts: the first part deals, very briefly, with problems where a performance criteria is to be minimised; while the second, more detailed, deals with problems where a cost function consisting of any combination of changeover, production and inventory costs is to be minimised. In both cases, at most one product is allocated to a given machine at any moment in time.

In the fourth chapter the overall approach for tackling the short term, detailed scheduling for the class of plants under consideration is introduced. In the first part of this chapter, a description of the plants considered is presented. Both structural and operational features are described. In the second part, the overall solution methodology is introduced. In particular, the coordination method between the two stages is described.

In the fifth chapter, the solution procedure for the packing lines subproblem is introduced. In this subproblem, the lot sizes of each item, output from the long term scheduler, are to be loaded onto the packing lines within the required time horizon, while minimising changeover and packing costs. Changeover costs may be sequence dependent and packing costs depend on the rate a particular item is packed on a particular packing line and other bottleneck considerations. Due to connectivity restrictions and to shared manpower resources, the lines are interdependent. The problem is formulated as a pure integer problem and a branch and bound algorithm is proposed.

In the sixth chapter, the production-intermediate storage problem is considered. The demand arising from the packing lines schedule is to be satisfied while minimising production and set-up costs. A tree search procedure that uses a Lagrangean relaxation of the original problem for computing lower bounds is proposed.

In the seventh chapter, some computational results are presented first for the packing lines algorithm, then for the overall procedure.

Chapter 2

Integer optimisation

2.1 Introduction

Integer optimisation problems are optimisation problems where all variables are required to be integer. When only some variables are required to be integer, the problem is termed a mixed integer optimisation problem.

Although integer optimisation problems are not new to mathematicians from the theoretical point of view, they have attracted much attention when it was recognised that many problems arising in the applications of operations research required integer variables in their models.

Integer optimisation problems arise naturally in real life situations. Indeed, in many real life situations, it is often required to answer one of the following questions: how many items, trucks etc... are to be chosen or allocated in order to minimise or maximise a certain objective? or which machine, route etc... is to be used in order to minimise or maximise an objective? Both these questions represent integer optimisation problems; in the latter case the variables can be coded to take the value of zero or one, for instance, a given variable is set to one if a certain route is chosen; in the former the variables may represent the number of items, trucks etc...

Historically, the development of methods for solving integer optimisation problems followed the development of linear programming. Indeed Dantzig had

shown [74] that there are integer optimisation problems that can be solved by linear programming (in particular the transportation problem). However the first convergent method for integer optimisation problem was the cutting plane method due to Gomory [29]. This was followed by other specialised or general methods; the most popular and widely used being the so called tree search methods or branch and bound or implicit enumeration methods, although, generally, the latter two are not always recognised to be synonymous. At the same time, specialised algorithms that use one or the other method were developed for particular problems. This was done mainly for problems like the travelling salesman, the knapsack problem or the assignment problem that were recognised to be standard problems, in that a number of real life situations can be described by these problems or by combinations of them.

In this chapter, a brief review of integer optimisation methods is presented, with an emphasis on tree search procedures and Lagrangean relaxation. In the first section a categorisation of integer programming problems based on the standard problems is presented. Also in the same section, methods of formulating integer problems are presented. In the second section, solution methods are briefly reviewed, while in the third a detailed description of tree search methods is presented. Finally the chapter ends with a section on Lagrangean methods as applied to integer optimisation.

2.2 Description of integer optimisation problems

2.2.1 Categorisation of integer problems

There are a few standard integer optimisation problems that can describe many real life situations. A categorisation of integer problems on this basis, was proposed by Muller-Merbach [61] who divided them into three categories: sequencing problems related to the travelling salesman problem, selection problems

related to the knapsack problem and assignment problems related to the linear assignment problem.

sequencing problems: in this type of problems, it is required to order all or a part of the elements of a set so that a certain objective is optimised. Among these problems one can identify the travelling salesman problem where it is required to find the optimal tour through a set of points starting from and returning to the same initial point, shortest path problems where it is required to find the shortest route between two different points and job sequencing problems where it is required to find the optimal sequence in which a number of jobs are processed on one or more machines.

selection problems: in these problems, it is required to select a number of elements from a set so that a certain return is minimised or maximised. These include the knapsack problem, the set partitioning problem and the set covering problem.

assignment problems: in this case, the elements of one set are to be assigned to the elements of another. These include the linear assignment problem, the general assignment problem and the quadratic assignment problem.

Many real life problems are combinations of these. However this categorisation is not unique, in that one can describe say sequencing problems as selection problems or assignment problems. Nevertheless it does give a good insight into the morphology of integer optimisation problems.

2.2.2 Formulation of integer optimisation problems

In formulating a problem one seeks ways of solution and insight into the problem structure. It is a general view that integer programming formulations are of great help in understanding integer optimisation problems, even if they are not used in developing solution procedures.

A linear integer programming formulation requires the definition of a linear objective function to optimise (for instance a cost to minimise or a profit to maximise) and the constraints to be satisfied by any solution to the problem. Among the constraints are included the integrality requirement on the variables.

It is worth pointing out, here, that any integer optimisation problem can be formulated as a zero/ one problem, that is a problem where the variables take only the two discrete values of zero and one.

In the following some examples of zero/one formulations are given for problems from the three categories introduced above.

Sequencing problems

In pure sequencing problems, the objective is to find the order of elements that minimises a certain cost. The constraints, generally, impose that every element has a predecessor, every element has a follower and no cycles exists. As an example, an integer programming formulation of the travelling salesman problem is presented here. This problem can be described simply as that of a person who wants to visit each of n cities exactly once, starting from a given city and returning to it, while minimising the total distance travelled.

To formulate the problem as an integer problem, let:

$$x_{i,j} = \begin{cases} 1 & \text{if city } j \text{ is visited after city } i \\ 0 & \text{otherwise} \end{cases}$$

Where $d_{i,j}$ is the distance between city i and city j with $d_{i,i} = \infty$

$$N = \{1, 2, \dots, n\},$$

the problem can be formulated as

$$\min \sum_i \sum_j d_{i,j} x_{i,j}$$

subject to the constraints:

1. every city must have only one following city

$$\sum_j x_{i,j} = 1 \quad i = 1, \dots, n$$

2. every city must have only one preceding city

$$\sum_i x_{i,j} = 1 \quad j = 1, \dots, n$$

3. One way of eliminating cycles [7] is to define S as any subset of elements of N and impose

$$\sum_{i \in S} \sum_{j \in \hat{S}} x_{i,j} \geq 1$$

where

$$\hat{S} = N - S$$

In the case of symmetric travelling salesman problem $d_{i,j} = d_{j,i}$

Selection problems

In pure selection problems, the objective is to select items or alternatives so that a certain return is maximised. In addition to the integrality constraints on the variables, there may be constraints on the number of items to be selected. As an example, an integer programming formulation of the knapsack problem is given.

In the knapsack problem, the objective is to select a number of items from a set to put in a limited capacity knapsack so that a certain return is maximised.

Letting

$$x_i = \begin{cases} 1 & \text{if item } i \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

the knapsack problem can be formulated as

$$\max \sum_i c_i x_i$$

subject to

$$\sum_i a_i x_i \leq b$$

$$x_i = \{0, 1\}$$

where c_i is the return accruing from selecting element i , b is the total capacity of the knapsack and a_i is the volume of element i .

Assignment problems

In the assignment problem, it is required to assign elements of one set to the elements of another. Depending on the type of problem, it may be possible to assign any element of one set to any element of the other or possibly to assign a subset of elements of one set to an element of another. With every assignment is associated a cost. The objective is to find the assignment that minimises the total cost.

A typical assignment problem is the *linear assignment problem*. In this problem, a one to one correspondence is imposed, that is every element of the first set must be assigned to one element of the second set and every element of the second set must have only one element of the first set assigned to it. Supposing that the elements of the first set are denoted by the subscript i and those of the second set by j , this problem can be formulated as follow:

Let

$$x_{i,j} = \begin{cases} 1 & \text{if element } i \text{ is assigned to } j \\ 0 & \text{otherwise} \end{cases}$$

The linear assignment problem can be stated as

$$\min \sum_i \sum_j c_{i,j} x_{i,j}$$

subject to

$$\sum_i x_{i,j} = 1, \quad j = 1, \dots, N$$

$$\sum_j x_{i,j} = 1, \quad i = 1, \dots, N$$

$$x_{i,j} = \{0, 1\}$$

Where N is the number of elements in the sets and $c_{i,j}$ is the cost of assigning i to j .

Another assignment problem is the so called *general assignment problem*. In this case, the N elements of a set A are to be assigned to the M elements of a set B . To every element of the first set is associated a weight and to every element of the second a capacity. Here, one element of the set B can have more than one element of the set A assigned to it or even none at all.

This problem can be formulated as follows

$$\min \sum_{i,j} c_{i,j} x_{i,j}$$

subject to

$$\sum_i x_{i,j} = 1, \quad j = 1, \dots, N$$

$$\sum_j a_{i,j} x_{i,j} \leq B_i, \quad i = 1, \dots, M$$

$$x_{i,j} = \{0, 1\}$$

The first constraint imposes that every element of the set A must be assigned to only one element of set B and the second constraint ensures that the total weight of the elements assigned to an element of the set B must not exceed its capacity.

A third assignment problem is the so-called *the quadratic assignment* which can be described as follows [47]: there are N plants and N locations, each plant is to be located on one location, and each location can take only one plant. The cost of carrying goods from location i to location j is $c_{i,j}$ and the quantity of

goods to be shipped from plant k to plant l is $b_{k,l}$. The problem is to find the allocation of plants to locations which minimises the total carrying cost.

Defining the variables:

$$x_{i,k} = \begin{cases} 1 & \text{if plant } k \text{ is located at } i \\ 0 & \text{otherwise} \end{cases}$$

The problem can be formulated as

$$\text{Min} \sum_i \sum_j \sum_k \sum_l c_{i,j} b_{k,l} x_{i,k} x_{j,l}$$

subject to

$$\sum_{k=1}^{k=N} x_{i,k} = 1 \quad \forall i$$

$$\sum_{i=1}^{i=N} x_{i,k} = 1 \quad \forall k$$

The objective function imposes that the cost $c_{i,j} b_{k,l}$ is incurred when plant k is located at i and plant l at j . The constraints ensure that one and only one plant is located at each location.

An alternative formulation for the TSP

Sometimes, because it is burdensome to formulate explicitly all the constraints of a problem, an implicit formulation is preferred. In the following, one such formulation using elements of graph theory is presented for the travelling salesman problem. This formulation was introduced by Held and Karp in [36]. The travelling salesman problem can be defined on a graph of n vertices, with V being the set of all vertices, and with a weight (cost) allocated to every arc connecting two vertices.

Before introducing Held and Karp formulation, however, the following definitions are needed.

A tree (an arborescence) is a connected undirected (directed) graph without cycles.

A spanning tree (arborescence) of an undirected (directed) graph is a tree (arborescence) connecting all vertices of the graph.

A minimum spanning tree (arborescence) of an undirected (directed) graph is a spanning tree (arborescence) of minimum weight.

Clearly, the problem of finding a minimum spanning tree in V is a relaxation of the symmetric travelling salesman problem. Moreover, defining a one-tree as a tree on all $N - 1$ vertices of V , which is connected to the starting node by exactly two edges, a minimum spanning one-tree is also a relaxation of the travelling salesman problem. The symmetric travelling salesman problem can therefore be described as the problem of finding the minimum one tree through the graph with the additional constraint that each vertex must have two edges incident to it. This gives the following formulation:

$$\min \sum_{i \in V} \sum_{j > i} d_{i,j} x_{i,j}$$

subject to

$$\sum_{i < k} x_{k,i} + \sum_{j > i} x_{i,j} = 2 \forall i$$

On a 1-tree.

Similarly the assymetric travelling salesman can be described using a shortest arborescence.

2.3 Solution methods for integer optimisation

In this section, methods for solving integer optimisation problems are briefly reviewed. Due to the discreteness of the values of the variables, the solution set of an integer optimisation problem is finite. Thus a straightforward solution procedure would be to enumerate all the feasible possibilities until the optimal solution is obtained. However, although this may be attractive for very small problems it becomes unrealistic for any real problem.

Two alternatives to explicit enumeration are available. The first consists of exact procedures, mainly integer programming techniques, although dynamic programming may be more efficient in some cases and for this reason should not be discarded *prima facie* . The second alternative consists of heuristic procedures.

In the following, a brief outline of these methods, including an overview of cutting planes techniques, dynamic programming, heuristic procedures and a more detailed survey of tree search procedures, is presented.

2.3.1 Cutting planes methods

Cutting planes [29] make an elaborate use of the simplex method, by adding to the current continuous solution a set of constraints, each representing a cut, that are violated by this solution but not by a feasible integer solution. The successive application of this process eventually gives an integer optimal solution. The solution is obtained only at the end of the procedure and no integer solution is available during the process. For this reason these methods are called dual methods. To alleviate this disadvantage, primal methods , that give integer solution in the course of the process have been developed [77]. Although cutting plane methods are attractive theoretically, it is generally recognised that they are not efficient in practice.

2.3.2 Tree search methods

Basically, all tree search methods consist of an exploration of the set of solutions of an integer optimisation problem. This exploration is done by dividing the problem into subsets and attempting to discarding some subsets from further exploration, either by proving that they do not contain any feasible solution or any optimal solution to the original problem or when an optimal solution is found. This is continued until no subset is left for consideration.

Since tree search procedures are the most widely used for solving integer

optimisation problems they are presented in more details in the third section of this chapter.

2.3.3 Dynamic programming

Dynamic programming was developed by R. Bellman [3]. It consists of a parallel enumeration of the solution set, based on what is called the optimality principle.

Optimality principle (Bellman) An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

In other words, the optimality principle states that given an optimal trajectory between two points A and C the portion of trajectory from any intermediate point B to C, must be the optimal trajectory from B to C. To be amenable to a dynamic programming solution, an integer optimisation problem must first be formulated as a multi-stage decision problem, consisting of say N stages $S_1, S_2, \dots, S_i, \dots, S_N$, such that there are, at every stage, alternative states T_1, T_2, \dots, T_P the system can take. The objective is to find the optimal path between the initial stage and the final stage.

The multistage decision process can be depicted as acyclic network, an example of which is shown in figure 2.1.

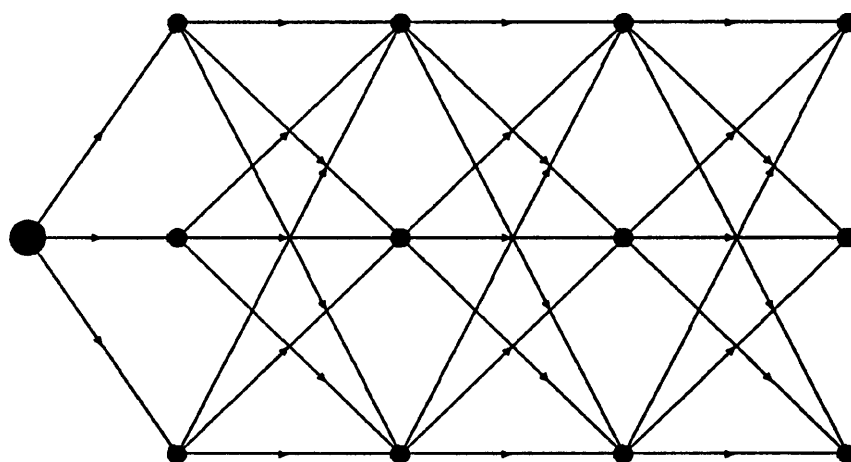
To find the optimal route (policy) the following recursive function is applied,

$$F_{i-1}(i) = \text{opt}_i [F_i(i) + c_i]$$

where *opt* denotes optimise.

This recursion gives the optimal subroute for every state. It is applied from the final stage to the initial stage. The optimal route is found by working forward from the initial stage to the final and selecting the optimal subroutes from one stage to the next. This process of working from the final stage to the initial stage is termed backward procedure. An alternative mode of recursion,

Figure 2.1: Acyclic network with 5 stages and 3 states



called forward procedure starts from the initial stage and evolves to the final stage. The optimal route is found by working backward from the last stage to the initial stage. In this case the recursive function can be written as

$$F_{i+1}(i) = \text{opt}_i [F_i(i) + c_i]$$

Dynamic programming is generally efficient for problems with deep and narrow networks, that is with few states and many stages.

2.3.4 Heuristics

Heuristic methods are often the last resort for integer optimisation problems. Indeed when none of the exact methods can produce a good feasible solution in a reasonable amount of time, one must settle for a speedily computable solution. However, in applying heuristics there is no assurance that the solution is optimal or even good, in any sense. Moreover one cannot be sure that the heuristic can give a feasible solution. Nevertheless because heuristic methods are fast, more than one method can be devised and the solutions obtained can be compared for selection.

There are two main classes of heuristics [61]: first feasible solution methods and improvement methods. First feasible methods start with an infeasible solution and construct a feasible one, using priority rules. Improvement methods start with a feasible solution and try and improve this solution gradually. Although the latter are generally preferred, it is not always possible to use improvement heuristics. Indeed, for some problems, it is already a difficult task to find a feasible solution.

2.4 Tree search procedures

Tree search techniques proceed by exploring the solution set of the problem. To this end, the solution set is gradually separated into smaller subsets, each corresponding to a subproblem of the original problem. Each time, an attempt

is made at discarding the subset under consideration, by showing that it does not contain an optimal or a feasible solution or by finding a integer feasible solution. If this attempt is not successful, the subset is further separated into smaller subsets and a new subset is selected for consideration. If the attempt is successful the problem is discarded and a new subset is selected. In the integer programming literature the process of discarding subsets is called *fathoming* and the last feasible solution called the *incumbent* [26].

From the above, it is apparent that the main questions which arise when applying tree search procedures can be summarised as follows:

- a)How is the separation of subproblems carried out?
- b)When is a subproblem fathomed?
- c)How is the subproblem to explore next selected?

In the following, answers to these questions are proposed, based mainly on Hansen [34]. The integer problem is supposed to be of the form:

$$(P) \quad v(P) = \min_x f(x)$$

Subject to

$$g(x) \leq b$$

$$h(x) \leq d$$

$$x \text{ integral}$$

2.4.1 Subproblem separation

Subproblem separation is carried out using a *branching rule*. Given a subproblem P_k with a set of candidate solutions C_k , and a set of feasible solutions F_k , a branching rule separates the set C_k into subsets $C_{i,k}$ such that

1.

$$\cup_i C_{i,k} = C_k$$

2. The number of the separations is finite.

3. When a subproblem can no longer be separated it must be fathomable.

Under these conditions, a branching rule is shown to converge. Generally, the initial problem consists of the original problem P which is separated into q subproblems that are stored in a list. A subproblem is then selected from this list and if it is not fathomed it is separated into subproblems that are added to the list. The process is terminated when the list of subproblems is empty.

2.4.2 Subproblems fathoming

The fathoming process is based on the concept of relaxation.

A relaxation of a problem P_k is a problem R_k defined in T_k , $C_k \in T_k$, with objective $r(x)$ and optimal solution $v(R_k)$ such that [26]:

1. $v(R_k) \leq v(P_k)$, that is the minimal value of R_k is always less than or equal to that of P_k .
2. If an optimal solution of R_k , $v(R_k) = r(x^*)$, is such that $x^* \in C_k$ then it is optimal in P_k , that is:

$$v(R_k) = r(x^*) = v(P_k) = f(x^*).$$

Given a relaxation R_k to a problem P_k , a number of tests can be constructed for fathoming this problem. The most important tests are optimality tests, solution tests and the feasibility tests. A description of these is given below.

Optimality test: given a relaxation R_k to a problem P_k and an incumbent z to the original problem P , if $v(R_k) \geq z$ then problem P_k is fathomed.

Feasibility test: a feasibility test is generally of the form:

Given a relaxation R_k to a problem P_k , if R_k has no feasible solution then P_k is fathomed.

Solution test: Given a relaxation R_k

and an incumbent solution z , if $v(R_k) = r(x^*)$, $x^* \in F_k$ and $v(R_k) < z$, z is replaced by $v(R_k)$ and problem P_k is fathomed.

Other, secondary, tests can be carried out, for instance the dominance test or reduction test. A dominance test is of the form:

Given two sets C_1 and C_2 , if it can be shown that the best solution in say C_1 is at least as good as the best solution in C_2 , then C_1 is said to dominate C_2 and C_2 is fathomed.

Reduction tests allow one to fix the value of some variable by showing that any other value they take will lead either to an infeasible solution or to a nonoptimal solution.

The choice of the relaxation is very important in a tree search procedure: the optimal value of R_k should be close enough to that of P_k and be fast to compute. However very often these two requirements are contradictory. Indeed, generally, the easier is R_k to solve, the greatest the difference between $v(R_k)$ and $v(P_k)$. A compromise, depending on the problem to be solved, must be sought.

The relaxations most widely used are the LP relaxation which is obtained by relaxing the integrality restriction on the variables and the Lagrangean relaxation which is presented in a following section.

2.4.3 Subproblem selection

After branching, the newly created subproblems are put in a list. The question arises as to which subproblem is selected next. There are two main rules for problem selection: the depth first rule and the breadth first rule. In the former, the last created subproblem is selected while in the latter a subproblem

satisfying a certain criterion is chosen; generally for minimisation problems, the subproblem with the smallest lower bound is selected.

The depth first rule requires a relatively small amount of storage, provides a feasible solution quickly and is easy to program. However, it has the disadvantage of being tedious when a subproblem is no longer separable, since in this case all the newly created subproblems have to be explored very often unrewardingly.

The breadth first rule requires much more storage and has the disadvantage of providing the first feasible solution late in the search. Hence, if the process is stopped prematurely, there may be no feasible solution. However, this rule tends to generate fewer nodes than the depth first when a good incumbent is available and a good relaxation is constructed. Commonly, in general purpose tree search procedures, the two options are available; the selection starts depth first and is switched to breadth first when a feasible solution is found.

2.5 Lagrangean relaxation

Lagrangean methods, well established in continuous optimisation were first introduced into integer optimisation as they are used today by Held and Karp [36] in their work on the travelling salesman problem. This work was followed by many other successful applications of Lagrangean methods to hard integer optimisation problems. Geoffrion in [26] developed the first systematic theory for the use of Lagrangean methods in integer optimisation and he coined the name Lagrangean relaxation.

In the following, Lagrangean relaxation is presented. The presentation is based on the work of Geoffrion [26] and the reviews by Shapiro [69] and Fisher [18].

2.5.1 The basic results

Let the problem be of the same form as problem P defined above. A Lagrangean relaxation for problem P is constructed by multiplying both sides of a subset of the constraints set by a non negative vector of multipliers λ if the constraints are inequality constraints of the form $g(x) \leq b$, by a non positive vector if the inequality are of the form $g(x) \geq b$ or by a real valued vector if the constraints are equality constraints. This subset is added to the objective function of problem P and deleted from the set of constraints, obtaining the relaxed Lagrangean problem (PR_λ) . This process is called dualisation of the constraints. If in problem (P) the constraints chosen to be dualised are $g(x) \leq b$ then the Lagrangean problem (PR_λ) is:

$$(PR_\lambda) \quad v(PR_\lambda) = -\lambda b + \min_{x \in X} (f(x) + \lambda g(x))$$

Where $X = \{h(x) \leq d \text{ and } x \text{ integral}\}$

The main properties of the Lagrangean relaxation are summarised in the following theorem:

Theorem 1 (Geoffrion) *Let PR_λ be a relaxation of P as defined above, then:*

1. $v(PR_\lambda) \leq v(P) \quad \forall \lambda \geq 0$
2. *If for a given λ a vector x^* satisfies the following optimality conditions:*
 - i) $v(PR_\lambda) = f(x^*) + \lambda(g(x^*) - b)$
 - ii) $g(x^*) \leq b$
 - iii) $\lambda(g(x^*) - b) = 0$

then x^ is optimal in P . If x^* does satisfy i) and ii) but not iii), it is said to be ϵ -optimal with $\epsilon = \lambda(g(x^*) - b)$.*

Proof.

For the first part of the theorem, by definition

$$v(PR_\lambda) \leq f(x) + \lambda(g(x) - b) \quad x \in X$$

then let \hat{x} be the optimal solution to P then

$$v(PR_\lambda) \leq f(\hat{x}) + \lambda(g(\hat{x}) - b) \leq v(P)$$

since $f(\hat{x}) = v(P)$ and $\lambda(g(\hat{x}) - b) \leq 0$.

To prove the second part, let x any feasible solution to P not satisfying the optimality conditions i), ii), iii) and x^* a feasible solution satisfying all optimality conditions, then for any $\lambda \geq 0$

$$v(PR_\lambda) \leq f(x) + \lambda(g(x) - b)$$

but $\lambda \geq 0$ and $(g(x) - b) < 0$ because x does not satisfy iii) thus

$$v(PR_\lambda) \leq f(x),$$

but $v(PR_\lambda) = f(x^*)$, for x^* satisfying i), hence

$$f(x^*) \leq f(x) \quad \forall x.$$

The first result shows that $v(PR_\lambda)$ can be used as a lower bound for $v(P)$, the second result that $v(PR_\lambda)$ can provide an optimal or a near optimal solution to P . The best bound one can obtain for P is given by the maximal value of $v(PR_\lambda)$ over all λ . This gives the following problem D :

$$(D) \quad v(D) = \max_{\lambda} v(PR_\lambda)$$

The following is a corollary of the above theorem:

Corollary 1 *If a pair λ^*, x^* satisfies the optimality conditions i), ii), iii) then λ^* is optimal in D .*

Proof.

$$v(D) = v(PR(\lambda^*)) = v(P)$$

$$v(PR(\lambda)) \leq v(P)$$

$$v(PR(\lambda)) \leq v(D)$$

From this, it can be seen that an equivalent to the optimality conditions is $v(D) = v(P)$. This means that when the optimality conditions are satisfied, there is no duality gap between $v(D)$ and $v(P)$. A duality gap may occur that it is $v(D) < v(P)$ and there is no couple λ^*, x^* satisfying the optimality conditions. This may happen because the dual problem (D) is the dual of the problem P^* defined on the convex hull of the set X . However, Lagrangean

relaxation is generally used for computing bounds and there is no attempt to find the optimal couple λ^*, x^* , although this may occur in the course of the computations. The current idea is to compute a vector of multipliers that provides a good $v(PR(\lambda))$ in the sense of approximating $v(P)$.

On the other hand, the ϵ -optimality suggests the use of Lagrangean relaxation for deriving good near optimal solution to P .

2.5.2 Computing the multipliers

Although the function $v(PR_\lambda)$ is continuous and concave, it is not differentiable everywhere. This means that optimisation methods for differentiable functions cannot be used in this context. One has then to resort to the methods of nonsmooth optimisation, particularly subgradient methods which, in fact, were developed as a byproduct of Lagrangean relaxation.

A subgradient is a generalisation of the notion of gradient and is defined as follows:

Definition

A subgradient of a function $v(PR_\lambda)$ at a point $\hat{\lambda}$ is a vector s such that:

$$v(PR(\lambda)) \leq v(PR(\hat{\lambda})) + (\lambda - \hat{\lambda})s$$

A function may have more than one subgradient at a point. A necessary and sufficient condition for local optimality (global for convex functions) at a point is that $\{0\}$ is a subgradient at that point.

A subgradient of the function $v(PR_\lambda)$ is given by

$$s = g(x_t) - b$$

where $x_t = \text{Argmin}_{x \in X} f(x) + \lambda g(x)$

Given a subgradient s of a function $v(PR_\lambda)$, the so called subgradient algorithm for non smooth optimisation generates a sequence of λ 's according to the

rule [64] :

$$\lambda^{k+1} = \lambda^k + t^k s^k$$

This rule converges to the optimal λ^* if the step size t^k satisfies the following:

$$\sum_0^\infty t^k = \infty$$

and

$$t_{k \rightarrow \infty}^k \rightarrow 0$$

The step size most widely used is of the form

$$t^{k+1} = \gamma^k (z - v(PR_\lambda^k)) / \|s^k\|$$

where $\|\cdot\|$ denotes the euclidian norm and

$$s^k = g(x_t^k) - b$$

The sequence satisfies the above conditions when $z = v(D)$, $\gamma^k = 2$.

However, the optimal value $v(D)$ is not generally known and the rule is to choose z as a good guess of $v(D)$ then choose $0 < \gamma^k \leq 2$.

Basically, the choice of γ^k is heuristic and depends on the choice of z . There is no assurance in using the subgradient algorithm that $v(PR_\lambda^{k+1}) > v(PR_\lambda^k)$ and a widely used rule is to halve the value of γ^k when $v(PR_\lambda^k)$ fails to increase for a number of iterations.

There are other recursion rules for generating a sequence of multipliers. Some are drawn from the theory of non smooth optimisation like the method of aggregated subgradients [46], which instead of using only the subgradient obtained at the k th iteration uses a linear combination of the subgradients obtained from say iteration k_0 to k . However these methods need the storage of more information than the subgradient algorithm and are certainly more time

consuming. Another method developed by Shor [70] related to the ellipsoidal algorithm and called the space dilatation algorithm works well but it also requires a large amount of storage space.

A good compromise may be to use the aggregated subgradient method combining only few subgradients obtained during the last, say two or three, iterations.

Other methods have been developed for particular cases like those reported in [19,59].

2.5.3 Lagrangean relaxation in tree search

When Lagrangean relaxation is used within a tree search procedure for computing bounds, generally only a few iterations of the subgradient algorithm are performed. The common rule [18,23,60] is to perform a large number of these iterations at the first node of the tree, obtaining an initial good lower bound, a small number of iterations at every forward step in the search and when backtracking perform an average number. In order not to perform unnecessary computations in forward steps, the multipliers are initialised to those corresponding to the best bound obtained in the previous step.

In addition to its use for optimality tests, Lagrangean relaxation is also used for solution tests. Indeed, as mentioned above, a feasible near optimal or an optimal solution to a subproblem can be found while computing the bounds. On the other hand, since the infeasibility of the Lagrangean problem PR_λ implies that of the original problem P , Lagrangean relaxation also provides feasibility tests.

The quality of the bounds obtained in Lagrangean relaxation depends mostly on the constraints dualised. Geoffrion showed in [25] that if the dualised constraints are such that the value of the Lagrangean is not increased when the integrality restrictions are dropped (a property he called the integrality property) then Lagrangean relaxation cannot do better than linear programming relaxation, but if this integrality property does not hold then the bounds ob-

tained are sometimes very sharp and better than those obtained from LP relaxation. However, for large scale decomposable problems Lagrangean relaxation is generally a better alternative than LP relaxation.

Chapter 3

Review of the relevant literature

3.1 Introduction

In this chapter, a review of the literature on production scheduling is presented. Due to the vast body of the publications on the subject, this review will be restricted to deterministic problems where at any time period a machine can process at most one product. This review is based on [14,24,31,53].

Before presenting the survey, it will be helpful to describe the setting in which production scheduling is carried out and to define its requirements and objectives. This is the subject of the second section. The third section presents a brief overview of scheduling problems involving the optimisation of a performance criterion. In the fourth section, a survey of the literature on problems more relevant to the purpose of this work is presented. These include problems in which a cost function is to be minimised. The cost can be a set-up cost, production cost, an inventory holding cost or any combination of these. The survey will cover single stage single facility, single stage multi-facility and multistage problems.

3.2 The production scheduling problem

Production scheduling is part of the production control system which encompasses a number of decisions that arise in the management of a company.

In the scheduling problem, the issues involved are the specification of the time and place at which events should take place. Problems of this sort arise in many environments, here only manufacturing environments will be considered. Thus an event will be the manufacturing (in a broad sense) of a product (job) and the place, the machine used to produce it.

There is a large variety of scheduling problems. These, usually, vary according to the mode of production, assumptions, constraints, requirements and optimising criteria. The mode of production can be an assembly mode, as in the car industry, or a process mode, as in chemical industry. Assumptions, constraints and requirements depend on the problem at hand. The optimising criteria can be classified into two categories: the first category is concerned with the optimisation of a performance measure; the second, with the optimisation of a cost function. In the first category one can name, for instance, the minimisation of the total makespan, the minimisation of the maximum tardiness or of the sum of tardiness, while in the second category the cost to be optimised can be any combination of changeover or set-up costs, production costs or inventory holding costs. The division adopted here will be along these lines. The following literature survey will be subdivided into two parts: the first part will be dealing with problems where a performance criterion is to be minimised and the second part with problems where a cost function is to be minimised.

The approaches used for solving scheduling problems are various and include graph theory, dynamic programming, linear programming, integer programming, network flows and heuristics.

3.3 Minimisation of a performance criterion

This category of problems is known as the job shop problem. The general job shop problem can be described as follows:

There are n jobs $1, \dots, j, \dots, n$ to be processed through m machines $1, \dots, i, \dots, m$. Each job has to be processed through the machines in a particular order and each job has its own processing order that may or may not depend on the processing order of the other jobs. The problem is to find a feasible schedule that is optimal with respect to a given optimality criterion.

Particular cases to the general job shop problem include the single machine problem, the parallel machines problem and the flow shop problem.

In the one machine problem, all jobs are to be processed once on one machine only. In the parallel machines problem, each job has to be processed once on any of a number of parallel machines. In the flow shop problem, all jobs are to be processed on the same set of machines with the same processing order (permutation schedules).

The jobs are usually characterised by a number of parameters. These include:

A processing time $p_{i,j}$ which is the time job j has to spend on the machine i . This may or may not depend on the machine.

A release date r_j on which the job becomes available for processing.

A due date d_j by which the job should ideally be completed.

A weight w_j indicating the relative importance of the job.

For a given problem a number of assumptions are generally made. These, usually, concern the following:

1. *Preemption*: preemption (job splitting) is or is not allowed,
2. *Precedence constraints*: there are or not precedence constraints among the jobs.

3. *In process inventory*: a job may or may not wait in inventory for its machine to be free .

The performance criteria are various and are generally function of one of the following:

1. *The completion time* of each job C_j , that is the time at which the processing of the job finishes.
2. *The lateness* of each job L_j , that is the difference between its completion time and its due date
3. *The tardiness* of each job T_j , which is the maximum between its lateness L_j and zero.

The most common performance criteria are:

1. Minimisation of the maximum completion time C_{max} .
2. Minimisation of the maximum lateness L_{max} .
3. Minimisation of the maximum tardiness T_{max} .
4. Minimisation of the mean completion time $\sum_j C_j$ or the weighted mean completion time $\sum_j w_j C_j$.
5. Minimisation of the mean lateness $\sum_j L_j$ or the weighted mean lateness $\sum_j w_j L_j$.
6. Minimisation of the mean tardiness $\sum_j T_j$ or the weighted mean tardiness $\sum_j w_j T_j$.

In general T_{max} , L_{max} , C_{max} are noted f_{max}

The relative difficulty of a problem depends on its category (single machine, parallel machine, flow shop or job shop). Within a category, the difficulty depends on the performance criterion. For a number of problems, especially in

the one machine category, it was possible to derive a number of constructive algorithms, whereby the optimal schedule is obtained by application of some simple rules. However, in general most algorithms are based on enumerative schemes. These features are illustrated in the following brief overview. This brief survey is mainly concerned with problems with no precedence constraints where the criteria are functions of the completion time. The emphasis will be on the constructive algorithms.

3.3.1 The single machine problem

The one machine problem is the simplest of the job shop problems and as such has attracted much attention. The best known result [53] is due to Jackson for the minimisation of L_{max} when all release dates are equal: the optimal schedule is obtained by scheduling all jobs in the order of non decreasing due dates. For the same criterion, when all due dates are equal, the optimal schedule is obtained by scheduling the jobs in the order of non decreasing release dates. For the minimisation of $\sum_j C_j$, the simple shortest processing time first rule (SPT) gives the optimal schedule whereas for the minimisation of $\sum_j w_j C_j$, the optimal schedule is obtained by Smith rule [71]: jobs are scheduled according to the non decreasing w_j/p_j .

Unlike the above, there are no constructive algorithms for the $\sum_j w_j T_j$ criterion and all the procedures are enumerative. A number of relaxations were constructed for the computation of lower bounds. In particular, one can cite the linear assignment relaxation of Rinnoy Kan [66] and the Lagrangean relaxation of Fisher [17], where the constraint that a machine should handle one job at a time is dualised. However, it seems that the most successful algorithm in this class is a well constructed dynamic programming algorithm due to Baker and Schrage [1].

3.3.2 Parallel machines

This category of problems is subdivided into the following three classes depending on the relationship between the machines:

Parallel identical machines : The processing time of a given job is the same on all machines $P_{i,j} = p_j$,

Parallel uniform machines : The processing time of a job on any machine is proportional to a certain constant: $p_{i,j} = p_j/q_i$,

Parallel unrelated machines No relationship exists between the processing times.

For nonpreemptive scheduling, constructive algorithms exist only for the minimisation of $\sum_j C_j$, whereas for the preemptive case there are also constructive algorithm for C_{max} . These results are reviewed below for the case where there are no precedence constraints.

Minimisation of $\sum_j C_j$

In this case, when the machines are identical, MacNaughton [57] showed that there is an optimal schedule in which no jobs are split. Thus, preemption is not an important issue for this problem. Conway et al have shown [53] that this problem can be solved with an algorithm similar to the shortest processing time rule. In this algorithm, the number of jobs is made a multiple of the number of machines: $n = km$ by adding dummy jobs with zero processing time. The jobs are classified according to increasing processing times: $p_1 \leq p_2 \leq \dots \leq p_n$. For $j = 1, \dots, k$, m jobs, $J_{(j-1)m+1}, J_{(j-1)m+2}, \dots, J_{jm}$, are assigned to m machines. Then, the k jobs assigned to each machine are loaded in shortest processing time order.

When the machines are uniform and preemption is not allowed, a generalisation of the above results for identical machines solves the problem. The problem with no preemption, when the machines are unrelated was formulated

as a transportation problem by Horn[38]. Defining the variable $x_{i,j,k}$ to be equal to one when J_j is scheduled on the k th position on machine i and zero otherwise, the problem can be formulated as follows:

$$\min \sum_i \sum_j \sum_k k p_{i,j} x_{i,j,k}$$

subject to:

$$\begin{aligned} \sum_i \sum_k x_{i,j,k} &= 1 \quad \forall j \\ \sum_j x_{i,j,k} &\leq 1 \quad \forall i, k \end{aligned}$$

When the machines are uniform and preemption is allowed, Gonzales constructed an algorithm where the jobs are first placed in SPT order [53]. An optimal schedule is then obtained by loading preemptively each job in the available time on the m machines so that its completion time is minimised.

Minimising C_{max}

As mentioned above, the only constructive algorithms for this class are for problems where preemption is allowed.

MacNaughton [57] gives a simple procedure for constructing the optimal schedule: the jobs are loaded onto the machines successively and in any order and a job is split whenever the following time bound is met:

$$\max(\max_j \{p_j\}, 1/m \sum_j p_j)$$

When the machines are uniform, Gonzales and Sahni generalised the MacNaughton procedure by showing [53] that the optimal value to this problem is:

$$\max \{ \max_{1 \leq k \leq m-1} \{ \sum_{j=1}^k p_j / \sum_{i=1}^k q_i, \sum_{j=1}^n p_j / \sum_{i=1}^m q_i \} \}$$

Finally, when the machines are unrelated, the problem can be modeled as a linear programming problem [53]. If $x_{i,j}$ is defined as the time job j spends on machine i the problem can be formulated as:

$$\begin{aligned}
& \min C_{max} \\
& \text{subject to} \\
& \sum_i x_{i,j} / p_{i,j} = 1 \quad \forall j \\
& \sum_i x_{i,j} \leq C_{max} \quad \forall j \\
& \sum_j x_{i,j} \leq C_{max} \quad \forall i \\
& x_{i,j} > 0
\end{aligned}$$

Before ending this section, it is worth noting that for the problem with criterion $\sum_j w_j C_j$ and no preemption, Eastman et al. [12] derived the following lower bound on the optimal solution:

$$\min(\sum_j w_j C_j) \geq m + n/m(n+1) \sum_{j=1}^n \sum_{k=1}^j w_j p_k$$

This bound was used in a number of branch and bound procedures.

3.3.3 The flow shop and job shop problems

Here, only a few milestones are cited on the very wide area of flow shop and job shop scheduling. In the flow shop problem, the criterion most widely optimised is C_{max} . The only constructive algorithm for this class is for the two machines problem. This algorithm was developed by Johnson [42] who showed that there exists an optimal schedule for the two machine problem in which job J_i precedes job J_k if:

$$\min\{p_{1,j}, p_{2,k}\} \leq \min\{p_{2,j}, p_{1,k}\}$$

Consequently, an optimal schedule can be constructed by finding at each step, among the unscheduled jobs, the smallest of the $p_{i,j}$, $i = 1, 2$, noted i_0, j_0 and loading the job j_0 onto machine i_0 .

All other algorithms for the flow shop problem are enumerative procedures. In branch and bound procedures, the most commonly used relaxation relaxes the constraint that each machine must handle one job at a time, for all machines but one [39], or two [49]. In the case where only one machine is unrelaxed, the bound computed from this relaxation is known as the machine based bound. Generally, elimination rules that eliminate non optimal schedules and dominance rules are very helpful [49].

As far as the general job shop is concerned there are no constructive algorithms and all the procedure are enumerative [16] [48]. It is worth noting that problems with 10 jobs and 10 machines have yet to be solved.

So far, only problems where the machines are independent have been considered. In real life, however, very often, this is not so. Indeed, there are scarce resources shared by the machines such as manpower or some special equipment. In such cases, generally, all machines cannot be operated simultaneously. This is the subject of the resource constrained scheduling which, for obvious reasons, is more difficult than the unconstrained scheduling. Noteworthy is the work of Sahney [68], who considers a problem with two machines with one server and proposes a branch and bound algorithm for solving it and the work of Stinson et al. [72] who propose a branch and bound algorithm for the multiple resource constrained problem. Davies [9] presents a survey of these problems.

3.4 Problems with cost function

The scheduling problem that will be considered here is a deterministic, dynamic problem, where at most one product can be allocated to any facility at any time period and in which a cost function is to be minimised. The survey will be started with the simplest problem of one product/ one facility and then extended

to the cases of multiproduct/ one facility , one stage multi-product/multi-facility and finally multistage problems.

3.4.1 One product one facility

This problem can be described as that of determining the amount to be produced of one product on one facility in each period, over a finite horizon so that a known demand is satisfied and a cost function consisting of set-up costs and/or production costs and/or an inventory costs is minimised. There may be constraints on the production and inventory capacities, in which case the problem is said to be capacitated. For this problem, the usual approach has been first to find some properties of the optimal schedule and then to examine only schedules having these properties. Commonly this has been done within a dynamic programming framework.

Such an approach, for the above model with no constraints on the capacities and no backlogging, was first considered by Wagner and Whitin [76]. They showed that in any optimal schedule, the demand, for each period, must be satisfied either from production or from storage but not from both. Also they derived the following inventory decomposition property:

Inventory decomposition property [76]

If inventory is zero at period k then it is optimal to consider periods 1, . . . , $k - 1$ by themselves.

Furthermore, they derived a planning horizon theorem which states basically that if it is optimal to incur a set-up in a period k when all periods up to k are considered by themselves, then optimality is not lost by letting production to be positive in k in the optimal solution for the overall problem. Thus, considering that it is optimal to produce at period k , it follows that it is optimal to consider periods from 1 to $k - 1$ separately.

These results allowed them to construct an algorithm that first finds the optimal policy for subproblems consisting of periods up to k , $k = 1, \dots, N$.

When all these policies are determined the optimal overall schedule is obtained by working backwards from the last period.

Every subproblem up to k is solved as follows:

First the policies of ordering at period t , $t = 1, \dots, k$ to fulfil the demand at $s = t, \dots, k$, are determined.

The cost associated with every policy is computed by adding to the cost of producing at t , the cost of acting optimally from 1 to $t - 1$. This latter was obtained by solving subproblem $k = t - 1$. Among all these policies the one with best cost is selected.

Zangwill [78] studied a similar problem with backlogging allowed and showed that the problem can be modelled as a single source network on which the arc flows are the production and the inventory. He derived the following properties for the extreme flows:

1. If the level of inventory is positive at $k - 1$ then the production is zero at k .
2. If production is positive at k then there can be no backlog.
3. If inventory is positive at $k - 1$ then there can be no backlog at k .

Then, defining a regeneration period α as a period where inventory is zero and a production period β as the next period after α where production is positive, he gave a recursion formula for computing the optimal cost from the beginning of a regeneration period $\alpha + 1$ to the last period in the horizon. He also derived a recursion for computing the optimal cost from the beginning of period β to the end of the last period. These recursion exploit the fact that an optimal schedule has the form of an extreme flow.

The calculations are performed backward from the last period, first the cost at a regeneration period is found. Given this cost, the cost at a production period is then computed.

Love [55] studied the case in which both inventory and production are capacitated, with both capacities allowed to vary in time. Defining a production

(inventory) point as a period in which production (inventory) is extreme, that is zero, maximum or minimum, he showed that there exists an optimal schedule which has the property that between any pair of production points (i, k) there is an inventory point j , $i \leq j < k$. From this, he developed an algorithm for the inventory capacitated case based on the converse of the above result, that is between a pair of inventory points there can be at most one production point.

He exploited the fact that at an inventory point cumulative production can equal exact requirement plus extreme inventory (zero, minimum or maximum) and gave an expression for the cost associated with augmenting the cumulative production between two inventory points i and k by producing at j , $i + 1 \leq j \leq k$, j production point.

In the case of a capacitated production problem, Florian and Klein [20] derived the following inventory decomposition property:

Inventory decomposition property [20]

If, for a T periods problem, at a period k inventory is zero and capacity is sufficient for the requirements at all $t \geq k + 1$ then the optimal solution can be found by considering the first k periods and then the remaining $k + 1, \dots, T$ periods.

This property allowed them to formulate the functional equation as

$$f_0 = 0$$

$$f_u = \min_{u,v} \{d_{uv} + f_v\}$$

where d_{uv} is the cost of following an optimal plan from $u + 1$ to v with $I_u = 0$ and $I_v = 0$, that is u and v regeneration points. For the case of constant capacity over time and no backlogging, they showed that d_{uv} is the cost associated with a shortest path through a certain network.

To do this, they first defined a constrained production sequence between two regeneration points as a series of time periods $t = u + 1, \dots, v$ where production is less than capacity in at most one period and is either at capacity or zero in

all others. They showed that any optimal schedule consists only of constrained production sequences. Thus, the search can be limited to constrained production sequences (uv) with cost d_{uv} . The network associated with d_{uv} has vertices corresponding to the values of cumulative production at period j , $j = u+1, \dots, v$. These values can be either 0, multiple of capacity or multiple of capacity plus a certain amount.

Extensions to the above basic models [76,78,55,20] were proposed by many authors. In particular, Swoveland [73] considered an uncapacitated problem with a piece-wise concave cost function, consisting of production cost and holding backorder cost. He developed a dynamic programming algorithm based on the work of Florian and Klein and that of Love.

A capacitated constrained problem with capacity variable with time was considered by Lambrecht and Vander Eecken in [50]. Using an approach similar to that of Florian and Klein, they showed that the optimal schedule between two regeneration points can be determined by the evaluation of a simple formula.

A slightly different framework from all the above was adopted by Baker et al. [2] who proposed a tree search algorithm for the time varying capacity problem. They first showed there is an optimal plan such that if there is inventory carried over from period $k-1$ to k , production at k must be either at capacity or zero; on the other hand if there is a positive production at a level less than capacity at k then the incoming inventory must be zero. They also showed that in an optimal solution, the last production quantity is either equal to capacity or to the sum of the remaining requirements.

Upon these results, a tree search algorithm was constructed. At every node of the tree the decision on which time period to place the last order is taken. For an N period problem, this is done sequentially as follows:

At the root of the tree the counter k_1 is set to $N+1$. N nodes are created, each corresponding to placing the last order at $k_1 = 1, \dots, N$, thus generating subproblems $P_1, P_2, \dots, P_{k_1}, \dots, P_N$. For each node the cost and the requirement that is not fulfilled by this order and thus must be fulfilled during $1, \dots, k_1 - 1$

are computed. If for $k_1 = 1$ the complete schedule is feasible it is stored as the incumbent. All subproblem created are stored in the list. In the same manner, every node corresponding to problems $P_2, \dots, P_{k_1}, \dots, P_N$, is further decomposed. For instance the node corresponding to $k_1 = 2$ generates one subproblem: ordering the sum of the unfulfilled requirement from subproblem P_2 and the requirement of period 1 at period 1, generating subproblem P_{12} . This subproblem giving a complete schedule, it is stored as incumbent if it is feasible and its cost is better than that of P_1 . Subproblem P_3 is decomposed into two subproblems P_{13} and P_{23} , corresponding to placing the last order at $k_1 = 1, 2$. Subproblem P_{13} corresponds to ordering the sum of the unfulfilled requirement from P_3 and requirements of periods 1 and 2 at period 1. This subproblem corresponds to a complete schedule and again if it is feasible and its cost better than that of P_{12} it replaces it as the incumbent. Subproblem P_{23} corresponds to ordering the unfulfilled requirement from problem P_3 plus requirement at period 2 at period 2, this subproblem which does not correspond to a complete schedule is stored in the list. Similarly subproblem P_4 generates P_{14}, P_{24}, P_{34} .

In general every subproblem, corresponding to placing an order at k_1 generates a number of subproblems corresponding to ordering at t the sum of the unfulfilled requirement from the father subproblem and all the requirements from t up to $k_1 - 1$. Every time $t = 1$ a complete schedule is obtained. If it is feasible and its cost is better than that of the incumbent it replaces it. If the problem does not correspond to a complete schedule it is stored in the list for further decomposition.

To limit the size of the tree a number of tests are applied. These check feasibility, optimality and the dominance of an uncomplete schedule over another.

3.4.2 Multi-product single facility

In this case there are more than one product sharing a single facility and the problem consists of scheduling all products over a finite horizon. At any time at most one product is allocated to a given machine.

Gilmore and Gomory [28] considered a one facility multi-products model with every product having one lot to be scheduled only once. The changeover between products were sequence dependent and it was required to find a sequence that minimises the changeover cost. The problem was transformed into a travelling salesman problem by adding a dummy product. The optimal solution will be given by the minimal cost tour starting and finishing at the dummy product.

In the approach adopted, the problem was first solved irrespective of the tour constraints, thus allowing an optimal solution to contain subloops. These subloops were eliminated by carrying out a number of interchanges. The interchanges were chosen by finding a minimal spanning tree in a connected graph.

A one machine multi-product finite horizon model was studied by Glassey [30]. The planning horizon was split into equal periods. The rates of production were the same for all products. A unit of production was set to what the machine can produce in one time period. Set ups were considered to occur between production periods.

In the model, a vector x was defined with every component specifying the cumulative production for each product and a network formulation of the problem was given. With each node of the network was associated a value of x . Two vectors x_1 and x_2 associated with two consecutive nodes were such that:

$$x_2 = x_1 + ke_i$$

with k integer and e_i a vector with all components equal to zero except the i^{th} entry which was equal to one. The length of any arc (x_1, x_2) was set to one.

This means that x_2 was the state of the machine obtained by producing k units of product i at the cost of one changeover. The distance between two nodes was defined as the shortest path between them.

The algorithm proposed can be described as follows:

Starting from the last period, with the cumulative requirements for every product, machine time is allocated to one product at a time, each time generat-

ing a new node. At time period t , $t - k$ consecutive time periods are allocated to product i , with k satisfying:

$$x_i(t - k - 1) < d_i(t - k - 1)$$

and

$$x_i(t - k) = d_i(t - k)$$

where $d_i(t)$ is the cumulative requirement for product i up to time period t .

In order to reduce the search, an optimality test was introduced.

Elmaghraby et al. [14] considered the scheduling of the lots of products in a single facility with each product to be produced only once in the horizon. The objective was to minimise a cost consisting of a set-up cost, an inventory cost and a backorder cost. The planning horizon was considered to be equal to the sum of set-up time and production time for all products (in the case where this is not true, a dummy product is added).

The problem was formulated as that of finding a shortest path through a network. With each node of the network was associated a state (S_k, h) where S_k represented the number of product yet to be scheduled and h the remaining time left in the horizon. Each node was decomposed into nodes corresponding to the production of a product in S_k and one node corresponding to the production of the dummy product.

A search algorithm similar to that of Glassey was developed. Starting from node (S_N, H) , when no product is scheduled, nodes were created according to the rule described above. At each node, a lower bound was computed consisting of the sum of the cost of the shortest path from the root node to this node and, the cost of scheduling the remaining products in the best possible way, irrespective of interference with other products. If the incomplete schedule thus obtained corresponds to a similar schedule obtained at another node, then the one with the larger lower bound is discarded.

To reduce the search a dominance test was constructed and precedence relations were derived. The dominance test was used to discard some nodes from further investigation by showing that they could not lie on an optimal path. The precedence relations were derived to show that if two products i and j were to be scheduled together then i must precede j .

Karmarkar and Shrage [43] considered a sequence dependent multi-product single facility problem with inventory, set-up and production costs. For every product, production was capacitated, with capacity varying from period to period. They studied a number of bounding procedures to be used in a tree search. First a relaxation of the coupling constraints, that is those constraints ensuring production only when set-up was carried out, was proposed. The problem was thus decomposed into two subproblems, one corresponding to the machine switching and the other with production and inventory, each with associated multipliers. However neither of this was solved; rather, they concentrated on the derivation of bounds for the problem with sequence independent set-up costs. For this they studied two Lagrangean relaxations.

The first was obtained by dualising the constraints ensuring that only one product is set on the machine at each time period. This allowed a decomposition of the original problem into dynamic single item, capacitated subproblems.

In the second, the same constraints were dualised. Then no longer considering capacitated production for every item but rather, a limited shared resource, they expressed the constraint ensuring the non utilisation of excess resource and dualised it as well. They showed that

$$w_1 \leq w_2$$

where w_1 is the bound obtained with the first relaxation and w_2 that obtained with the second.

Computational results using the latter relaxation were not encouraging.

3.4.3 Multi-product multi-facility

Dorsey and al. [10] considered a multi-product multi-facility problem where the facilities are identical, in that a product can be processed in any facility with the same rate and cost, the demand is known for every period in the horizon. Set up are assumed to occur between production periods.

At any time period for a given product either there is no production or production is a multiple of a fixed amount corresponding to what one facility can produce in a time period. The cost function consists of production costs and inventory holding costs. The production costs are time independent. The requirements are two fold: first, demand must be satisfied and second, a desired end inventory must be built for every product.

The fact that at any time period, for a given product, either there is no production or production is a multiple of a fixed amount, corresponding to what one facility can produce in a time period, allowed them to give a simple integer formulation of the problem.

They defined an integer parameter $w_{i,k}$ that determines for every period k and every product i the minimum number of times product i must be produced so that inventory at k is nonnegative and an integer variable $x_{i,k}$ which gives for every period k the number of facilities allocated to product i . They first expressed the total inventory cost in terms of the average inventory built in one time period. Then, noting that when at the end of the horizon there must be exactly a required amount of inventory the production cost is the same in any solution, they formulated the problem in terms of inventory costs only. This allowed them to give a network flow representation of the problem, with nodes k , ik and a source. Flow from source to node k indicates the total number of facilities used at k , with the upper bound representing the total number of facilities available. Flow from k to ik gives the number of facilities allocated to product i at k . Flow from ik to $ik + 1$ gives the number of times product i has been produced up to k with lower bound $w_{i,k}$.

Further, remarking that it is more economic to build the required inventory

in the last periods, they developed an optimal one pass procedure, based on priority rules. Starting from the last period, they allocated machines to products such that priority is given to products with higher inventory cost.

In [56] Love and Vegumanti studied a multi-products multi facility problem arising in the tyre industry. The requirement was to produce at least a minimum amount of every product during every period of the horizon. The production capacities of the machines were limited. The problem was to minimise sequence independent set-up cost only.

They first gave an integer formulation of the problem with integer variables being the number of set-ups and removals for every product at each period and the number of product on machines at every time period. Then introducing a supplementary variable to account for idle machines (in their case empty cavities) they transformed the constraints matrix into a totally unimodular matrix, allowing the problem to be formulated as a minimum cost flow problem. Flows in arcs correspond to the number of set-up take-downs and number of machines on and off.

A different model, where the facilities were not identical, was considered by Prabakhar [65]. Every product could be produced at different rates on different facilities with different costs but at most once on a given facility in the horizon considered. The set-ups were sequence dependent and there were inventory limitations. The demand for every product occurs at the end of the horizon and the objective function to be minimised consists of set-up and production costs.

The problem was formulated as a mixed integer program with real valued variables for the quantities produced of each product on every machine and $(0, 1)$ variables for the changeovers between product. Only constraints preventing subloops of order two were included in this model, that is loops containing two products. The solution, therefore, was allowed to contain either a loop, one or more subloops or a disconnected loop.

A commercial code was used for the solution. If the solution contains loop

or subloops, it is reoptimised using a shortest path algorithm. The method, thus, gives a suboptimal overall solution. In order to improve the result the procedure could be repeated.

Geoffrion and Graves [27] studied a similar problem to Prabakhar, with demand occurring at every time period and no inventory limitation. The possibility of splitting the production of a product into more than one run on the same machine was allowed. Assuming that the production rates on all machines and all changeover times were proportionally related, they introduced a quantisation of the model. They defined time slots and production lots. The time slots correspond to a partition of the scheduling horizon. The total number of time slots was equal the number of time slots in the horizon multiplied by the number of machines. The production lots correspond to a partition of the production orders for every product, each lots having an earliest start time and a latest finish time. This allowed them to express the problem as a one to one mapping from the lots to the slots , with the one to one correspondence obtained by adding a dummy product when necessary.

The cost function included a changeover cost, a production cost and a prohibitive penalty for scheduling a lot before its earliest start time or after its latest finish time. They used a quadratic programming procedure for solving the problem. Clearly the problem defined in this way can be too inflexible. To avoid this, they included a linear programming procedure for smoothing the production orders.

Parker et al. [63] studied a multi-product, multi-facility problem with identical facilities and sequence dependent changeover costs without preemption. The problem was to find a feasible schedule with minimum changeover costs. They modelled the problem as a vehicle routing problem and used a well-known heuristic to solve it. Taking advantage of the speed of this algorithm, they developed an iterative procedure where at every iteration a new vehicle routing problem with new constraints that eliminate previous solutions is solved. This is done until no more improvement can be achieved.

3.4.4 Multistage models

The layout of the facilities in a multi-stage manufacturing system depends on the particular application.

In general a serial system is a system where each facility has one predecessor and one follower, except the final one which has only a predecessor and the first one which has only a follower. This type of model can be met in the chemical industry. In an arborescence system, a facility has one predecessor and one or more followers except the first and the final facilities. In an assembly systems, a facility can have any number of predecessors or followers. Typically, these systems are met in the car industry.

A number of optimal procedure have been proposed for one product multi-stage systems. A brief review of these is given below.

The general approach taken is akin to the single facility single stage problem, that is a characterisation of the optimal schedule is sought in order to limit the search to schedules having the properties derived.

In particular, Zangwill [78] studied a one product serial model with all facilities un-capacitated, known demand and concave production and inventory costs for all facilities. The problem was formulated as a single source network flow. Using some properties of extreme flows, he showed that at any facility production takes place only if there is no entering inventory. Kalyon [44] extended this property to the arborescence system. For a similar problem to that of Zangwill with non decreasing production cost over time and inventory cost non decreasing over the facilities, Love [54] showed that if in a given period i production starts at a facility j , then all successors of j produce at i . He called a schedule exhibiting this property, a "nested schedule".

Crowston and Wagner [6] extended these two properties to general assembly systems.

Lambrecht et al [51] studied a model where the last facility was capacitated. They showed that if at the last facility there is positive inventory carried from previous periods, then production is either zero or at capacity and if there is

production at less than capacity then inventory must be zero.

Unfortunately most of the optimal procedures are too time consuming and thus impracticable in real life. Therefore a number of heuristic methods were developed. There are mainly two type of heuristics: multi stage heuristics and single stage heuristics.

In single stage heuristics, one stage problems are solved sequentially. Starting with the last stage, production schedules are determined for every stage. The production schedule for a given stage determines the demand for its predecessor. Each stage is solved either with its original cost or with a modified cost.

This modification is carried on the set-up cost or on the inventory cost. In case of the modified set-up cost, two approaches were proposed. Letting the set-up cost at facility j be s_j , these approaches can be summarised as follows:

In the first approach, called cumulative set-up, the modified cost at facility j is computed as

$$s_j^* = \sum_1^j s_j$$

While in the second it is computed as follows:

$$s_j^* = s_j + As_{j-1}$$

with A a constant.

In multi stage heuristics, production quantities are assigned to all facilities period by period. A multi stage heuristic was proposed by Lambrecht et al. [51]. They defined a reorder period as the last period with positive production at all facilities. For each facility and each time period a coefficient is computed indicating whether a cost reduction is possible by incorporating demand from a later period than the reorder period, in a lot that has already been scheduled in earlier period. If all coefficients are negative for a period greater than the reorder period, then this period is considered as the new reorder period and the

procedure is repeated. In the above heuristics every stage is solved once and no overall optimality is attempted. These heuristics are sometimes called single pass heuristics. Lambrecht et al. [51] present a comprehensive study of these heuristics.

Graves [32] proposes a more elaborate multi pass approach. In this approach, the problem is solved iteratively, at every iteration the subproblems corresponding to all stages are solved. The solution is improved at each iteration and coordination is achieved using a marginal cost. For this, a variable cost of production $p_{j,t}$ is defined for each facility j at every time period t . At every iteration this cost is revised by incurring a marginal cost $\gamma_{k,t}$ of increasing demand for production at the predecessor k of j by one unit in period t . The new cost $p_{j,t}^*$ is given by:

$$p_{j,t}^* = p_{j,t} + \sum_{k \in P_j} \beta_{k,j} \gamma_{k,t}$$

where

P_j indicates all predecessor of j ,

$\beta_{k,j}$ is the number of units that facility k must produce for each unit produced at j , and

$$\gamma_{k,t} = p_{k,t} + (t - t_0)h_k$$

where t_0 is the last period of production for k . In other words, if demand in period t is increased by one unit, production in period t_0 increases by one unit with an increased production cost at t_0 plus a cost of holding product in inventory from t_0 to t .

3.5 Summary

As is apparent from this survey, there is a wide variety of scheduling problems and methods for tackling them. However, due to the combinatorial nature of most of these problems, particularly those where preemption is not allowed, enumerative methods are very often the only resort. The methods developed

exploit the particularities of the problem at hand in order to improve the efficiency of the algorithm. Most models assume the availability of all resources. This is not so in real life situations where there are always scarce resources. This, often, leads to mismatches between scheduling theory and practice.

Chapter 4

Problem description and solution methodology

4.1 Introduction

In this chapter, the overall solution procedure adopted for the short term production scheduling for a class of plants which produce a series of detergents is presented. As mentioned earlier, the study carried out here is concerned with the second level of a two-level hierarchical approach.

In the methodology adopted for the production planning and scheduling problem, the higher level is concerned with the long term decisions while the lower level is concerned with the short term decisions. At the higher level, the long term scheduler is based on aggregate data: the items are grouped into families and the demand and resources are aggregated into aggregate demand and aggregate resources. The costs involved are set-up and inventory holding costs. For a long term horizon, made of say N short periods, the long term scheduler gives the quantities (lot sizes) to produce of each family, for each of the N short periods. The lower level is basically concerned with a two-fold disaggregation: an item disaggregation which determines lot sizes for every item individually (as opposed to the aggregate planning which gives the lot sizes for aggregate families) and a time disaggregation or short term scheduler.

Based on actual data, for a short term made of T small time periods, the short term scheduler allocates machine time and other resources as raw materials and manpower, for the production of the lot sizes of each individual item. All lot sizes must be produced within the short term. --

The problem studied here is concerned only with the latter issue of the lower level. However before presenting the problem, the general features of the manufacturing plant are described in detail in the next section. In the third section, the short term commitment is introduced while in the fourth section the overall solution methodology is presented.

4.2 Description of the plants

4.2.1 General features

Most of the plants considered here are two-stage production processes: upstream, there are parallel production facilities where base products are manufactured; down stream, there is a number of parallel packing lines where the base products are packed in different formats. Between these two processes, there is an intermediate storage system that feeds the packing lines generally through a number of material handling systems. Figure 4.1 shows the topology of a typical plant with two manufacturing units, four storage silos, three material handling systems and four packing lines.

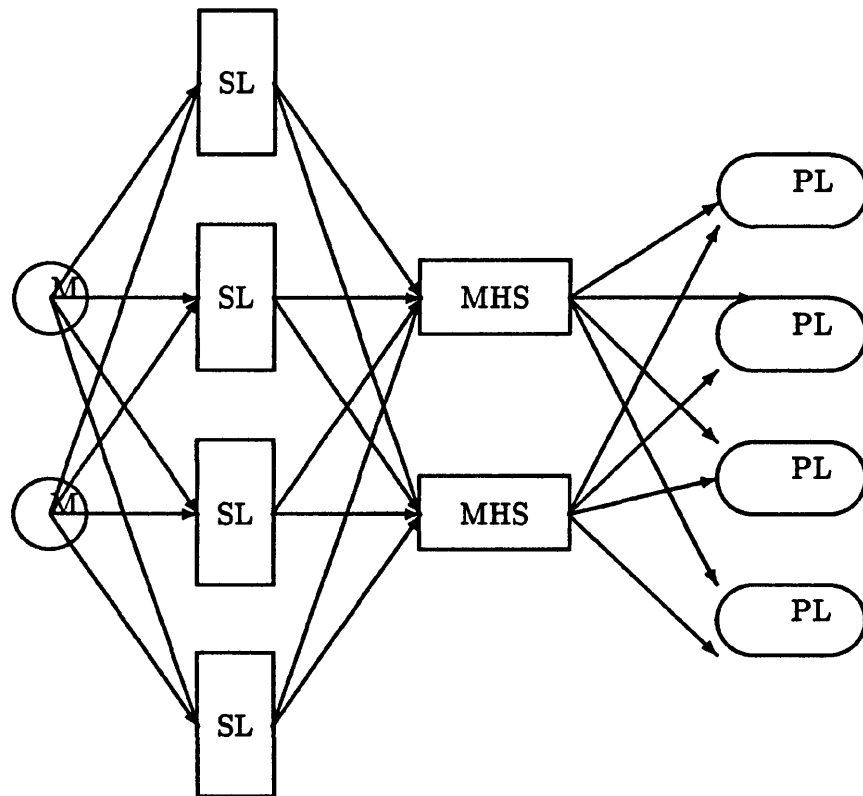
In the following, each of these entities is described.

4.2.2 Manufacturing units

There are, commonly, from one to four manufacturing units which can make from one to ten products, one at a time. After production, the products are either stored temporarily before packing or directly fed to the packing lines through the material handling system.

The units are highly automated. For the units that are dedicated to the

Figure 4.1: A typical lay out of a two stages plant



manufacturing of a single product, cleaning is not generally necessary. Cleaning is very important in the case of the units that make more than one product (general purpose units). This is mainly due to undesirable interaction between products, such as colour interaction and cross contamination. Cleaning time is of the order of one hour. Long runs are very desirable to keep the packing lines packing with minimum changeover and to minimise the manufacturing units changeover costs. In some plants, the units are operated two or three shifts of eight hours a day, in others they are operated one shift a day only.

4.2.3 Intermediate storage

The intermediate storage consists of silos that can either be dedicated to one product or flexible. Flexible silos can store any product, one at a time. In the latter case cleaning is necessary when changing over from one product to another. The capacity of the flexible silos depends on the density of the product stored.

The silos are fed from the manufacturing units through a series of conveyors or pipes. Some units may not be connected to some silos and there may be restrictions on the number of connections that can be active at the same time. There are maximum flow constraints on the connections but, usually, these do not cause bottlenecks.

The silos in turn feed the material handling systems through pipes or conveyors and there can, also, be restrictions on the number of simultaneously active connections.

4.2.4 Material handling systems

The material handling systems feed the packing lines and can handle one product at a time. There can be a maximum of seven material handling systems. On the other extreme there can be none, that is the packing lines are fed directly from the silos. Again the number of simultaneous connection from the material

handling systems to the packing lines can be restricted. Maximum flows on the connections do not cause bottlenecks. The material handling systems can be fed directly from the manufacturing units.

4.2.5 Packing lines

There can be from five to ten packing lines that can pack one or more formats (sizes). Some are dedicated to packing the different formats of one product while others can pack different products in different formats. A combination size/product will be called an item. Packing rates depend on both the item and the line. There are no minimum or maximum packing rates, a line packs an item at a fixed nominal rate. Although it will be assumed throughout that the lines pack at their constant nominal rate once switched on for an item, start up rates are generally slower and a line may take several minutes if not hours to reach full efficiency.

There are two types of changeovers: minor changeovers and major changeovers. Minor changeovers occur between items of the same size, are sequence independent and do not usually require intensive labour. Major changeovers occur between sizes, may be sequence dependent and require intensive labour by skilled fitters. They take longer to carry out than minor changeovers.

The packing lines are operated by operators whose number depends on the item the line is packing. This number can vary from one to forty depending on the line and the item considered. Usually there are from ten to twenty operators in the packing room. This of course limits the number of simultaneously active packing lines. Packing lines operations are further constrained by the incoming connections from the silos or/and from the material handling systems. Also, the number of products that can be packed at the same time in the packing room is restricted by the number of material handling systems, if any.

In some plants, the packing lines are operated for two shifts of eight hours a day, in others, due to expensive night operations only one daily shift.

4.2.6 Planning cycle and demand

The planning cycle varies from six to ten weeks according, in particular, to the location of the plant and the brands produced. The ideal situation for the company would be to have a fixed plan of production that can be used every cycle. However this is not possible due to equipment breakdown, promotional sales and demand uncertainty.

Demand is usually made of a series of orders with earliest and latest due dates for delivery. The size of the demand depends on the particular item, for some items the seasonal demand is low whereas for others it is very high.

4.3 The short term commitment

At this level, the major costs are introduced by the packing lines operations. These costs are two-fold: changeover cost and packing costs. Changeover costs are mainly associated with the major changeovers while packing costs are associated with the rates a line packs a particular item and some bottleneck considerations.

At the manufacturing level, long runs are very desirable to allow for long packing runs and to minimise the number of set-ups for the manufacturing units.

Finally, a good schedule should use the intermediate storage efficiently and also minimise the number of changeovers for the flexible silos.

In this problem, it is required that at the end of the horizon, of one or two weeks, the lot sizes of every item, output from the upper level, should be produced.

There are no black box methods for finding the optimal solution for a two stage multi-product manufacturing system with intermediate storage. A monolithic job shop formulation does not seem to be fruitful. Indeed, the resource constraints, the intermediate storage and the minimum run length requirements make the problem overconstrained. For the multistage systems the approaches

proposed are generally heuristic. These heuristics consider the problem stage by stage or period by period. Generally most heuristics are one pass [33], in that every stage is considered once, exceptionally they can be multipass: the problem is solved iteratively until a sufficiently good solution is obtained.

For these complex problems, the main issue is feasibility, specially in situations where resources are scarce. Typically, scarce resources overconstrain the scheduling problem, making it difficult to load all the lot sizes in the available time horizon.

For the problem considered here, a one pass heuristic is impracticable since there is no guarantee that a feasible schedule for one stage would give a feasible schedule for the other. This is due to the fact that, on the one hand, there are, generally, several products sharing the same manufacturing facility and on the other hand the operating speed of the packing lines is different from that of the manufacturing units.

As an example, consider a problem with 2 items, 2 packing lines, one manufacturing unit and a horizon of three periods of one hour each. The two items are of different base product and size, each with a lot size of three tons. The packing rates are, for both items, one ton per hour on both packing lines and the production rate is, also for both items, ten tons per hour (these figures correspond to a real life situation). Supposing the optimal schedule for the packing lines is the following:

line 1: (hour 1,item 1)(hour 2,item 1)(hour 3,item 1),

line 2: (hour 1,item 2)(hour 2,item 2)(hour 3,item 2),

it can, immediately, be seen that unless there is a sufficient initial amount of either of the two products in the intermediate storage there are no solutions to the problem at the manufacturing units. Therefore, this packing lines schedule has to be reconsidered. Imposing that both lines must pack the same product during the first period, will give, among others, the following schedule

line 1: (hour 1, item 2) (hour 2, item 1) (hour 3, item 1)

line 2: (hour 1, item 2) (hour 2, item 2) (hour 3, item 1).

Clearly, this schedule will result in a feasible manufacturing unit schedule.

Now consider another example with 1 manufacturing unit, 3 packing lines, 3 items of the same product with same packing rates of 4 tons/hour on all lines. The demands are 8, 12 and 10 tons for items 1, 2 and 3 respectively. Initial inventory is zero and the production rate is 10 tons/hour. Suppose that the optimal schedule is the following:

line 1: (hour 1, item 1) (hour 2, item 1) (hour 3, item 1)

line 2: (hour 1, item 2) (hour 2, item 1)

line 3: (hour 1, item 3) (hour 2, item 3) (hour 3, item 3),

it is readily seen that there is no feasible schedule at the manufacturing unit level since the demand at the first period is greater than the production capability of the manufacturing unit. If the number of items packed in period 1 is reduced from 3 to 2 by delaying the packing of item two, a feasible schedule can be obtained. These remarks will be used for developing the overall solution procedure.

4.4 Solution methodology

4.4.1 Overall approach

A stage by stage, multipass procedure has been adopted. As mentioned above, overall feasibility is the main issue.

The packing lines subproblem is solved first. The lot sizes output from the first level are scheduled so that a cost function consisting of a combination of changeover and packing costs is minimised. The basic time period can be anything from one shift to one hour. All the lot sizes must be packed during the horizon; sometimes, however, overtime is available.

The schedule obtained is then input as a demand for the manufacturing units. At this stage, the problem is to satisfy this demand, minimising a cost function consisting of set-up and production costs; the latter being either the costs of processing a particular product on a particular unit or the costs of

manufacturing at certain periods (night shifts for instance). Due to dimensionality considerations, the basic time period can be no less than half a shift. Both subproblems are solved using enumerative procedures. If there are no feasible solutions to the manufacturing units problem, the whole problem is reconsidered. But before presenting the coordinating device between successive iterations and in order to give some insight into how a bottleneck situation could be alleviated, the following considerations are introduced.

In the above examples, it is clear, that there are two possible ways of achieving feasibility: reconsidering the initial inventory for one or more products or altering the demand at the manufacturing units level by altering the quantity packed at a certain time period. The second possibility may overconstrain the packing lines problem, leading to infeasibility, the first depends on the previous schedule of the manufacturing units. Infeasibility can also be alleviated either by introducing some overtime at the manufacturing units or at the packing lines level or by reducing some lot sizes. Finally it is worth noting, that due to the aggregate nature of the data, the lot sizes output from the long term scheduler may not give a feasible solution at the lower level. In such a case, it is necessary to alter some of the lot sizes.

4.4.2 The coordination method

The coordination device adopted is a heuristic, interactive method, in that the user can choose which variables to alter and how to alter them.

As mentioned above, one way of alleviating infeasibility is to alter the demand at the manufacturing units level. A possible way of varying the quantities packed by a line at a time period, and altering the demand at the manufacturing units, would be to associate some shadow prices on these quantities and build an iterative procedure based on these prices. However, here, this is not feasible since the packing lines are packing at fixed nominal rates. This implies that at any time period, a line will either pack the quantity it can pack at this time period or be idle. Therefore, one way of altering the total quantity packed at a

given period, of a given product, in the packing room, is to reduce the number of items, belonging to this product, that can be packed in this time period. Thus, forcing one or more packing lines to be idle or packing another product.

On the other hand, if the number of products to be manufactured at a given time period exceeds the number of manufacturing units, then the maximum number of products that can be packed at this time period can be reduced. Also adding, directly, a positive increment to the initial inventory and solving again the manufacturing units subproblem may improve feasibility. Since the overall problem is solved on a rolling horizon basis, the previous schedule of the manufacturing units is known and hence the maximum quantity of initial inventory that can be built.

The procedure developed is based on the above remarks. When it is necessary overtime will be added. All parameters can be input manually.

Before presenting the algorithm, the following variables are defined:

$P(t)$: the maximum number of products that can be packed simultaneously in the whole packing room at period t ,

$I(j, 0)$: the initial inventory for product j ,

$PI(j, t)$: the maximum number of items belonging to the product j that can be packed during time period t ,

$PA(t)$: the number of products packed at time period t in the current packing lines schedule,

$PIA(j, t)$ the number of items of product j , packed during t in the optimal packing line schedule.

In the following, it is supposed without loss of generality, that the length of the basic time period is the same in the manufacturing units subproblem as in the packing lines subproblem.

Step0. Initialise

$$r = 0, P(t) = P^0(t), PI(j, t) = PI^0(j, t), I(j, 0) = I^0(j, 0).$$

Step1. Solve the packing lines subproblem with $P^r(t)$, $PI^r(j, t)$.

If a feasible schedule is found go to step3.

Step2. The packing lines schedule is infeasible, two alternatives are offered:
either extend the horizon by adding an increment Ot :

$$T = T + Ot$$

where Ot is fixed by the user.

and go to step1, or alter the previous packing lines schedule to reduce demand on the manufacturing units.

Step3. Solve the manufacturing units subproblem, with $I^r(j, 0)$ and the demand arising from the packing lines schedule.

If a satisfactory, feasible schedule is found stop.

Step4. Infeasible schedule.

Update either $PI^r(j_t, t_b)$ or $P^r(t_b)$ and go to step1. or

Step5. The manufacturing units subproblem is to be reconsidered with new initial inventory. Choose one or more products say j_i , $i = 1, s$ for which initial inventory is to be increased, set

$$I^r(j_s, 0) = I^{r-1}(j_s, 0) + ic \quad \forall j_i. \text{ and go to step 3.}$$

The total increment added should not exceed the possibility of building initial inventory. This possibility can easily be assessed given the previous manufacturing units schedule.

Remarks

1. The way $PI^r(j_t, t_b)$ or $P^r(t_b)$ will be updated will be presented in the sixth chapter, after introducing the solution methodology for the manufacturing units-intermediate storage subproblem.
2. Step5. can only be performed with external intervention.

Chapter 5

Short term commitment for the packing lines

5.1 Introduction

In this chapter, the scheduling of the items lot sizes on the packing lines is considered.

The lot sizes of each item are to be loaded onto the packing lines so that changeover and packing costs are minimised. The packing lines share some resources in common. A branch and bound algorithm is proposed.

This chapter is organised as follows: in the second section, the packing lines problem is described. In the third section, the problem is formulated as a $(0, 1)$ integer linear program where pre-emption is allowed. Due to the size of the resulting problem, a transformation is introduced, allowing a more tractable formulation where preemption is not allowed. In the fourth section, a relaxation of the new problem is proposed and bounds are computed. This relaxation decouples the problem into two subproblems : a machine loading problem, formulated as a general assignment problem, and a pure sequencing problem, formulated as a shortest spanning arborescence problem. Finally, in the fifth section, the solution algorithm together with a heuristic for computing upper bounds is described.

5.2 Description of the system

The overall manufacturing system was described in chapter 3. Here, the relevant characteristics of the second stage of this system are recalled.

At this stage, the manufactured products are packed in various formats in the packing room. Some packing lines are dedicated to a particular size, some to a particular product and some are general purpose in that they can pack many products in different sizes. Packing rates vary from one pack size to another and for a particular product/pack combination (item) from one line to another. The changeovers between items may be sequence dependent but do not differ from line to line. The operations of the packing lines are restricted by manpower and structural constraints:

- **Structural constraints:** The lines are fed with base products either directly from intermediate storage or by up to ten material handling systems. These are, in turn, either fed from intermediate storage or are connected directly to the manufacturing units. Each material handling system can carry one product only at a time, thus restricting the total number of products that can be packed simultaneously by the whole packing system. The number of simultaneously active connections from one material handling system to the packing lines is restricted, in general, to two or three.
- **Manpower constraints:** A number of operators is usually needed for operating the packing lines. For a given line, the number of operators depends, generally, on the item it is packing. Changeovers from one size to another are executed by skilled fitters, whose number varies according to the sequence considered. Usually, minor changeovers between items of the same size do not need skilled fitters.

In most plants, the packing lines are generally operated for fifteen or ten shifts of eight hours a week.

Due to the sometimes significantly slow start up rates, it is desirable to keep the lines packing a given item for a minimum number of shifts before changing over to another item. This constraint is usually termed the minimum run length constraint.

For short term scheduling, usually over a period of one week, lot sizes are given by the long term scheduler, and the objective is to find a feasible combination of items/packing lines so that high packing rates and minimum changeover times are obtained. All lots have to be produced before the end of the horizon. It must be pointed out, that due to the aggregate nature of the upper level, there is no guarantee that this would be possible. Sometimes overtime must be needed so that all lot size are loaded.

The problem has the features of a machine loading problem and those of a sequencing problem at the same time. However, a further complication is introduced by the fact the lines are interdependent, due both to the shared manpower resources and the structural constraints. Hence, not only one has to find the optimal items/lines combination, but also to consider the schedule at discrete time periods in order to handle the resource constraints efficiently.

It is clear that the choice of the length of the time step at which decisions are taken is critical, since this choice will determine the degree of flexibility and the size of the problem. If it is too small, the number of variables will be prohibitively large. On the other hand if it is too large, there will be a loss of precision.

Some authors considered a similar problem to the one described above, but without the resource constraints (see chapter 3). In particular, a problem with sequence dependent set-up costs and non identical machines was considered by Prabakhar in [65]. A mixed integer formulation was proposed and a branch and bound procedure adopted. Geoffrion and Graves [27] also studied a similar problem and used a quadratic assignment algorithm to solve it. Parker et al. [63] studied a case with identical machines and sequence dependent set-up costs and proposed a solution based on an algorithm for the vehicle routing problem. Love

and Vegumanti [56] considered a model with identical machines and sequence independent set-up costs and proposed a network flows formulation.

It is clear that none of the above procedures is applicable here. Indeed, the features that allowed one or the other formulation are lost by the introduction of the resource constraints. In any case, the introduction of these constraints in the basic model will lead to a formidable problem. To show this, in the next section a $(0, 1)$ integer formulation of the unrestricted problem is introduced.

5.3 A unrestricted problem formulation

To give an idea of the complexity of the problem, in the following a $(0, 1)$ integer formulation is proposed. The following assumptions are introduced:

1. Any item must be packed just once on a given line during the current horizon.
2. A changeover is carried out within one time period. Thus, a time period must be at least equal to the longest changeover time.
3. The last item packed by each line during the last horizon is known. If in the current horizon there is demand for this item it will be numbered differently. A zero changeover cost will be introduced between these two items.
4. Start up rates are not taken into account.

The discrete formulation imposes that at any time period a line is either packing at full capacity, being changed over or idle. Thus, if the line is packing at a given time period the quantity packed is simply equal to the amount the line is capable of packing within one time period.

Let the lines be numbered $i = 1, \dots, L$, the items with positive demand be numbered $j = 1, \dots, N$ (subscripts k and l are also used to identify these items) and the time periods in the horizon be $t = 1, \dots, T$. Defining the following variables

$$\alpha_{i,j,t} = \begin{cases} 1 & \text{if line } i \text{ is packing item } j \text{ during period } t \\ 0 & \text{otherwise} \end{cases}$$

$$\delta_{i,j,l,t} = \begin{cases} 1 & \text{if line } i \text{ is being changed-over from packing item } j \\ & \text{to packing item } l \text{ during period } t \\ 0 & \text{otherwise} \end{cases}$$

and given

d_j Demand (lot size) for item j

$ct_{j,l}$ cost of changeover from j to l

$p_{i,j}$ cost of packing item j on line i

$r_{i,j}$ packing rate of item j on line i per time period.

np_j number of operators required for packing item j

$nf_{j,l}$ number of fitters for changeover from j to l

NO total number of operators

NF total number of fitters

MHS Number of material handling systems

MRL_j Minimum run length, in time periods, for item j .

k_i Last items packed on line i during the previous horizon, numbered outside N (fictitious items).

P Number of base products

C_1 relative weight of the total changeover cost.

C_2 relative weight of the total packing cost.

the problem can be formulated as

$$(P) \min C_1 \sum_i \sum_j \sum_t p_{i,j} \alpha_{i,j,t} + C_2 \sum_i \sum_j \sum_l \sum_t ct_{j,l} \delta_{i,j,l,t}$$

Subject to

$$\sum_t \sum_j \alpha_{i,j,t} + \sum_t (1 - \sum_j \alpha_{i,j,t}) + \sum_t \sum_j \sum_l \delta_{i,j,l,t} \leq T \quad \forall i \quad (5.1)$$

$$\sum_i \sum_t r_{i,j} \alpha_{i,j,t} = d_j \quad \forall j \quad (5.2)$$

$$\sum_j \alpha_{i,j,t} + \sum_j \sum_l \delta_{i,j,l,t} \leq 1 \quad \forall i, t \quad (5.3)$$

$$\alpha_{i,j,t_1} \leq \sum_{t=1}^{t_1} \sum_k \delta_{i,k,j,t} - \sum_{t=1}^{t_1} \sum_{l=1} \delta_{i,j,l,t} \quad \forall i, j, t_1, t \quad (5.4)$$

$$\sum_{t=t_0+1}^T \alpha_{i,j,t} \geq MRL_j \sum_k \delta_{i,k,j,t_0} \quad \forall i, j, t_0 \quad (5.5)$$

$$\sum_i \sum_j np_j \alpha_{i,j,t} \leq NO \quad \forall t \quad (5.6)$$

$$\sum_i \sum_j \sum_l nf_{j,l} \delta_{i,j,l,t} \leq NF \quad \forall t \quad (5.7)$$

$$\sum_j \sum_t \delta_{i,j,l,t} \leq 1 \quad \forall i, l \quad (5.8)$$

$$\sum_l \sum_t \delta_{i,j,l,t} \leq 1 \quad \forall i, j \quad (5.9)$$

$$\sum_i \sum_j \sum_t \delta_{i,j,k_i,t} = 0 \quad \forall k_i \quad (5.10)$$

$$\sum_j \sum_t \delta_{i,k_i,j,t} \leq 1 \quad \forall k_i \quad (5.11)$$

$$\sum_i \sum_j \sum_t \delta_{i,j,j,t} = 0 \quad (5.12)$$

$$\delta_{i,j,l,t}, \alpha_{i,j,t} = \{0, 1\} \quad (5.13)$$

where

- Constraints (5.1) ensure that the sum of the packing time, the idle time and the changeover time should not exceed the total time available.
- Constraints (5.2) impose that the demand should be satisfied exactly for each item.
- Constraints (5.3) impose that at any time period, a line must be packing at most one item, idle or undergoing at most one changeover.
- Constraints (5.4) ensure that a line packs an item only if it has been set and stops packing it if it has been changed over.
- Constraints (5.5) ensure that a line packs an item at least for the minimum run length.
- Constraints (5.6) ensure that the number of operators, needed at every time period, does not exceed the total available.
- Constraints (5.7) ensure that the number of fitters, needed for executing all the changeovers at time period t , does not exceed the number of fitters available.

- Constraints (5.8) impose that every item is set once on a given line
- Constraints (5.9) impose that an item is changed over once on a given line.
- Constraints (5.10) ensure that the fictitious items are not set on any line and (5.11) ensures that they can have at most one follower.
- Constraint (5.12) ensures that no item is changed over from itself.

A few remarks are in order here. Clearly, due to (5.4) and (5.11), a cycle containing any k_i cannot occur. Constraints (5.4), (5.5), (5.8), (5.9), (5.11), (5.12) eliminate other cycles. To see this, consider a cycle starting and ending at item j . Such a cycle implies:

$$\delta_{i,j,k,t} = 1 \text{ and } \delta_{i,l,j,t^*} = 1$$

where $t < t^*$

To show that this can not happen, consider the minimum run constraints (5.5) for item j :

$$\sum_{t^*+1}^T \alpha_{i,j,t} \geq MRL_j$$

meaning that there must be a run for j . Due to constraint (5.9) and (5.4), this run cannot occur before t^* . It must occur after t^* . Substituting in (5.4) gives:

$$\alpha_{i,j,t^{**}} = 0 \quad \forall t^{**} > t^*$$

Thus, there could not be a run after t^* . Therefore when there is a cycle starting and ending at j , there will be no run for j , implying that constraints (5.4) and (5.5) are violated.

On the other hand, supposing that an item satisfies the minimum run constraints, it can readily be seen that it cannot start and end a cycle. Indeed, if a given item j is set at time period t_0 , i.e.: $\delta_{i,k,j,t_0} = 1$, the minimum run length gives

$$\sum_{t_0+1}^t \alpha_{i,j,t} \geq MRL_j$$

implying $\alpha_{i,j,t} = 1$ for $t > t_0$. Substituting in (5.4) and using (5.8) and (5.9) gives:

$$1 < \delta_{i,k,j,t_0} - \delta_{i,j,l,t}$$

which means that $\delta_{i,j,l,t} = 1$ implies $t_0 < t$. In other words, if item j was set at t_0 then it is changed over at $t > t_0$.

This formulation is valid only when the changeover times are all of the same order and small, say between 3 hours and eight hours, thus allowing one to take a time step of something like eight hours without a big loss in precision and flexibility. However, when the differences between changeover times are large, another formulation, where a changeover can take more than one time period, is necessary.

So far structural constraints have not been included. To do so, entails defining a new variable

$$\theta_{i,j_p,t} = \begin{cases} 1 & \text{if any item of product } j_p \text{ is packed on line } i \\ & \text{during period } t \\ 0 & \text{otherwise} \end{cases}$$

Then, for example, the material handling constraints can be written as:

$$\sum_i \sum_{j_p} \theta_{i,j_p,t} \leq MHS \quad (5.14)$$

One can see that the problem as formulated above is intractable. Indeed, the large number of 0,1 variables and constraints makes it beyond the capabilities of any existing method for integer programming.

It is, therefore, necessary to carry out some transformation such that the resulting problem is feasible, tractable and its solution constitutes a good, near optimal solution to the original problem. In the following section one such a transformation is proposed. The new, more tractable formulation allows for a feasible solution to be developed speedily. The approach has the following features:

- The solution of the restricted problem is carried out through a specialised tree search scheme.

- The computation of bounds used to limit the search is based on a relaxation of the linking constraints, that is those constraints that link the sequencing part to the loading part of the problem.
- The resource constraints are handled separately.

5.3.1 A problem transformation

An important specification for the problem is that any solution must satisfy the minimum run length requirement. However, this specification is imposed explicitly on the model at the upper level of the problem (long term scheduler). Thus, all lot sizes are equal to a multiple of the minimum run length plus a certain quantity, ie:

$$d_j = kMRL_j + \epsilon$$

Therefore, it is unnecessary to specify explicitly these constraints at the lower level. Instead, it would be sensible to split every lot q_j into k smaller lots of MRL_j each or $k - 1$ of MRL_j each and one of $MRL_j + \epsilon$. In general, each lot size can be split in a number of smaller lots each representing a new item to be considered on its own. As an example suppose that the lot size for a given item is approximately of five tons and the minimum run length for this item is of two tons. This lot could be split into two small lots, one of two tons and the other of three tons. Every small lot will have to be scheduled without splitting (nonpre-emptive scheduling). It is worth emphasising that this transformation is not severe in terms of optimality. In the above example, it can readily be seen that the feasible solutions are either one run of five tons, two runs of two tons and a half each or one run of three tons and the other of two. If $V(Pr)$ is the optimal solution to the transformed problem Pr and $V(P)$ the solution to the original problem then clearly: $V(Pr) \geq V(P)$. However, because of the above remarks $V(P)r$ is very close and often equal to $V(P)$. Moreover, because it is

more convenient to handle the resource constraint separately, hereafter, they will not be included in the formal description of the problem.

5.4 Relaxation and lower bounds for the transformed problem

In this section, the method for computing the lower bounds for the transformed problem is introduced. After formulating the problem without the resource constraints, a relaxation is proposed and algorithms for solving the subproblems arising from this relaxation are constructed.

5.4.1 Relaxation

Dropping the resource constraints, introducing the transformation proposed above and defining the following variables:

$$x_{i,j} = \begin{cases} 1 & \text{if lot } j \text{ is packed on line } i \\ 0 & \text{otherwise} \end{cases}$$

$$\beta_{i,j,l} = \begin{cases} 1 & \text{if lot } j \text{ precedes lot } l \text{ on line } i \\ 0 & \text{otherwise} \end{cases}$$

The given:

$a_{i,j}$: packing time of item j on line i ,

$ch_{j,l}$ changeover time between items j and l ,

the problem can then be formulated as

$$(Pr1) \quad \min C_1 \sum_i \sum_j p_{i,j} x_{i,j} + C_2 \sum_i \sum_j \sum_l ct_{j,l} \beta_{i,j,l}$$

subject to

$$\sum_i x_{i,j} = 1 \quad \forall j \tag{5.15}$$

$$\sum_j a_{i,j} x_{i,j} + \sum_j \sum_l ch_{j,l} \beta_{i,j,l} \leq T \quad \forall i \quad (5.16)$$

$$x_{i,j} - \sum_l \beta_{i,l,j} = 0 \quad \forall i, j \quad (5.17)$$

$$\sum_i \sum_l \beta_{i,l,k_i} = 0 \quad \forall k_i \quad (5.18)$$

$$\sum_i \sum_k \beta_{i,j,k} \leq 1 \quad \forall j \quad (5.19)$$

$$\text{No circuits} \quad (5.20)$$

$$x_{i,j}, \beta_{i,j,l} \in \{0, 1\} \quad (5.21)$$

Where the parameters are the same as before and subscripts i designated the packing lines, j, l designate the items to be scheduled in the current horizon and k_i designate the fictitious items. Constraints (5.15) stipulate that an item is to be packed on one line only; constraints (5.16) ensure that the total load on any line must not exceed the time available; constraints (5.17) ensure that if an item j is loaded on line i , then it must follow an item l on this line; constraints (5.18) impose that all the first (fictitious) items must have no predecessor and constraints (5.19) impose that all items can have at most one follower.

Clearly, the solution to $Pr1$ is a lower bound on Pr , the original transformed problem. However, solving $Pr1$ to optimality within a tree search algorithm is time consuming as this will involve the solution of a large problem at every node of the tree. Therefore, in computing bounds for Pr only a relaxation of $Pr1$ will be considered.

If the linking constraints (5.17) are relaxed and the changeover times neglected, problem $Pr1$ can be decomposed into two subproblems: a machine loading problem with the constraint that every lot has to be packed on one line only and a sequencing problem. The machine loading subproblem is given by:

$$(Pr11) \quad \min C_1 \sum_i \sum_j p_{i,j} x_{i,j}$$

Subject to (5.15) and (5.16) with the changeover time neglected. However, in the algorithm developed, an empirical penalty (which is a lower bound on the changeover times) is introduced to account for the changeover time.

The sequencing subproblem is

$$(Pr12) \quad \min C_2 \sum_i \sum_j \sum_l c_{j,l} \beta_{i,j,l}$$

Subject to equations (5.18), (5.19), and (5.20), (5.21).

The last problem, in its general form when the last items are not known, can be transformed to an M-traveling salesmen and it could be solved so. However, instead, a relaxation of *Pr12* will be constructed, leading to a lower bound (*lb3*) which will be fast to compute. The computation of these lower bounds will be presented in subsection 1. 4. 3.

Calling the optimal solutions to (*Pr11*), *lb1*, that to (*Pr12*), *lb2*, the above leads to:

$$LB = lb1 + lb3 \leq lb1 + lb2 \leq V(Pr1) \leq V(Pr)$$

LB being a lower bound on the optimal solution of problem *Pr*. At every node of the tree developed for the solution of *P*, both *Pr11* and the relaxation *Pr12* are solved, giving the bound *LB*. In order to improve this bound, penalties will be computed. Before introducing the global algorithm developed for solving *Pr*, in the following, the methods adopted for solving *Pr11* and the relaxation of *Pr12* and for computing penalties are presented.

5.4.2 The machine loading subproblem

Problem *Pr11* corresponds to a general assignment problem and can be rewritten simply as:

$$(Pr11) \quad \min C_1 \sum_i \sum_j p_{i,j} x_{i,j}$$

Subject to

$$\sum_i x_{i,j} = 1 \quad \forall j \quad (5.22)$$

$$\sum_j a_{i,j} x_{i,j} \leq T_i \quad \forall i \quad (5.23)$$

$$x_{i,j} = \{0, 1\} \quad (5.24)$$

Where T_i is the time left for packing on line i at the current node of the search tree less a penalty to account for the changeover times.

This problem can be efficiently solved by a branch and bound algorithm using a Lagrangean relaxation [18,4,67] of constraint (5.22). This approach is followed here.

Dualising constraint (5.22), gives the Lagrangean problem:

$$(LRG) \quad L(\lambda) = \min \sum_i \sum_j p_{i,j} x_{i,j} + \sum_j \lambda_j (1 - \sum_i x_{i,j}) \quad (5.25)$$

Subject to (5.23) and (5.24). From now on the factor C_1 will be omitted.

It is well known that:

$L(\lambda) \leq lb1 \quad \forall \lambda$, i.e. $L(\lambda)$ is a lower bound on $lb1$. The best lower bound will be

$$LG_\lambda = \max_\lambda L(\lambda)$$

One may have $LG_\lambda = lb1$, but this is not guaranteed, due to the fact that LG_λ corresponds to the dual of the linear program obtained from $Pr11$ by dropping the integrality constraints.

In the solution procedure, $L(\lambda)$ is used as a lower bound to the optimal value of the subproblem generated at a given decision node. No attempt is made at finding the λ for which $L(\lambda) = lb1$, although this may occur, in which case the optimal solution to the subproblem at the current node is obtained.

Equation (5.25) can be rewritten as :

$$L(\lambda) = \sum_j \lambda_j + \min_{x_{i,j}} \left(\sum_i \sum_j (p_{i,j} - \lambda_j) x_{i,j} \right)$$

$$L(\lambda) = \sum_j \lambda_j - \max_{x_{i,j}} \left(\sum_i \sum_j (\lambda_j - p_{i,j}) x_{i,j} \right)$$

With $x_{i,j}$ subject to (5.23), (5.24).

When the multipliers λ are fixed, this problem decomposes into L independent single knapsack problems [67], the solution of which is used in a subgradient algorithm to maximise the augmented objective function over λ . The new fixed values of λ are used to update the knapsack problems.

The knapsack problems

For every packing line i , the following knapsack problem KP_i has to be solved:

$$\max \sum_j (\lambda_j - p_{i,j}) x_{i,j}$$

Subject to

$$\sum_j a_{i,j} x_{i,j} \leq T_i$$

A branch and bound algorithm with the reduction tests constructed by Nauss [62] has been adopted for solving KP_i . All the details on the solution of knapsack problem are given in appendix A.

The algorithm for solving Pr11

A depth first binary tree search for the solution of Pr11 was implemented. At any node of the tree, a binary variable, corresponding to loading an item onto a line, is set to one and the remaining variables corresponding to loading the same item onto the other lines are fixed to zero. Thus, a level of the tree consists of a set of brother nodes representing different loadings of a given item.

In their approach to the general assignment problem, Ross and Soland [67] suggest that a good bound is obtained when the multipliers are set to the second best cost for every item. However, in the case where the minimum cost for each

item occurs on more than one line, the bounds thus obtained are very low. Indeed, suppose that for an item j :

$$\min_i p_{i,j} = p_{i_1,j} = p_{i_2,j}$$

then if the multiplier λ_j is set to the second best cost for this item, that is:

$$\lambda_j = \min_{i \neq i_1, i_2} p_{i,j}$$

the contribution of this item to the Lagrangean, noted L_j , might be:

$$L_j = \lambda_j - ((\lambda_j - p_{i_1,j}) + (\lambda_j - p_{i_2,j}))$$

and

$$L_j = p_{i_1,j} + p_{i_2,j} - \lambda_j$$

Thus, L can become nonpositive if $\lambda_j \geq 2p_{i_1,j}$. Of course the situation will become worse if there are more than two lines where the item has the best cost. Nevertheless, in the algorithm developed, the multipliers were set to the second best costs. However, as soon as the subgradient associated with an item becomes negative, control is transferred to the branching rule so that the item is loaded on one line only, thus strengthening the bound. The iterations of the subgradient algorithm are then resumed with the same multipliers. This is done until no item has the subgradient associated to it negative. The procedure is then repeated for the unloaded items. The multipliers are set to the second best among the lines where the item is still loadable. A node is fathomed if an item becomes unloadable.

Likewise, in a backward step, each multiplier is set to the second best cost among the lines where the associated item is loadable.

In the following the best feasible line, in terms of cost $p_{i,j}$, for item j is denoted by i_j and the second best by i_{jj} .

Step0. Initialisation

Step0a. Set $t'_i = t_i$ where t_i is the time left up to the end of the horizon, for line i , at the current node of the overall algorithm.

For all unscheduled items, noted j , form the sets

$$I_j = \{i \mid p_{i,j} < \infty \text{ and } a_{i,j} \leq t'_i\}$$

For all these items, set $x_{i,j} = 1, i_j \in I_j$.

If the solution thus obtained is feasible in *Pr11*, then it is optimal, stop;
otherwise

Step0b. Set the Lagrange multipliers to:

$$\lambda_j = p_{i_{jj},j} \quad \forall j \quad i_{jj} \in I_j$$

$$\text{set } LG^* = \sum_j p_{i_{jj},j}$$

$$lb1^* = \infty$$

$$s = 0$$

$$P_s = \text{Pr11}.$$

Step1. Lower bounding

Step1a. Optimality test

Solve the Lagrangean relaxation of P_s with the current vector of multipliers λ .

If $LG \geq lb1^*$ then go to step4;

Step1b. Feasibility test

If the solution to the relaxed problem is feasible in the dualised constraints,
update LG^* and $lb1^*$ accordingly, store the solution obtained and go to
step 4.

Step2. Branching step

Step2a. Selecting the item

In the solution to the last Lagrangean problem, the following will occur:

$$1 - \sum_i x_{i,j} < 0 \text{ for some } j$$

$$1 - \sum_i x_{i,j} = 0 \text{ for some } j$$

$$1 - \sum_i x_{i,j} = 1 \text{ for some } j$$

Look for a j_b such that

$$\lambda_{j_b}(1 - \sum_i x_{i,j_b}) = \min_j \lambda_j(1 - \sum_i x_{i,j}) \text{ and } 1 - \sum_i x_{i,j} < 0.$$

If there is none, then if there is a j such that $I_j = \emptyset$ go to step4, otherwise initialise the vector of multipliers to $\lambda_j = p_{i_{jj},j}$ and go to step1.

Step2b. Selecting the line

Among those lines satisfying the following condition in the current Lagrangean solution

$$\sum_j a_{i,j} x_{i,j} < t'_i$$

choose the one for which p_{i,j_b} is minimum, call this i_b , if there is none, choose $i_b \in I_{j_b} \mid p_{i_b,j_b} = \min_i p_{i,j_b}$, go to step 3.

Step3. Forward step

Set $s = s + 1$, $x_{i \neq i_b, j_b} = 0$, $x_{i_b, j_b} = 1$ in P_s ,

$t'_i = t'_i - a_{i_b, j_b}$ and $R_s = (i_b, j_b)$. Go to step1.

Step4. Backward step

Step4a. If $s = 0$ then the incumbent is the optimal solution and stop; otherwise

Step4b. Extract i_b and j_b from R_s .

Set $t'_{i_b} = t'_{i_b} + a_{i_b, j_b}$

Update the set $I_{j_b} : I_{j_b} = I_{j_b} - i_b$.

If $I_{j_b} \neq \emptyset$ then set the initial λ to

$$\lambda_j = p_{i,j,j}$$

go to step2b, with the extracted j_b ; otherwise $s = s - 1$ and go to step4a.

5.4.3 The sequencing subproblem

In this section, the computation of a lower bound to the sequencing part of the problem is presented. As remarked above, this problem is somewhat simplified by the fact that the state of the lines at the end of the previous horizon is known.

It is worth emphasising that this is a realistic assumption because in the situation dealt with here, rolling horizons are considered and in any case the scheme developed for the solution of the overall problem allows this assumption to be included easily. Indeed, at every node of this scheme the state of the lines is known hence, the initial assumption holds at all nodes.

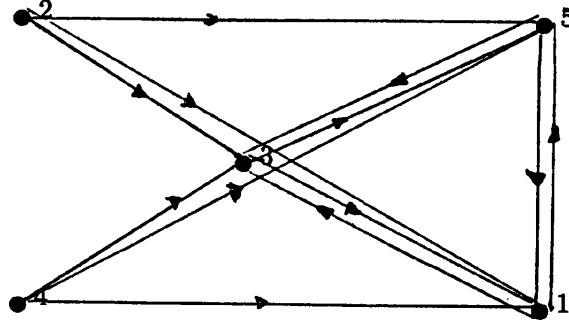
At any node, the sequencing problem can be modeled as a directed graph rooted at the state of the system at this node. The root of the graph will correspond to the items being packed by the lines and spanning through the remaining items. This can easily be seen in the following example.

Consider there are two lines and five items, and suppose that at the current node line 1 is packing item 2, line 2 is packing item 4, and items 1, 3, 5 are to be sequenced. The corresponding graph is shown in figure 5.1.

A lower bound on the changeover cost can be obtained by finding the shortest arborescence through the graph. Polynomial algorithms for constructing the shortest arborescence of a graph have been devised by Chu and Li [5], Edmonds [13] and Fulkerson [21]. These algorithms are basically the same. In [75], Tarjan gave a $\mathcal{O}(n^2)$ version of the algorithm of Chu and Li and Edmonds. A version of the algorithm of Fulkerson has been implemented, since it fits in the formulation of the sequencing subproblem.

First, all items being packed on all lines at the current node k are grouped in one component \mathcal{R} . Then let \mathcal{X} be the set of vertices consisting of the items not

Figure 5.1: Directed graph with two lines and five items



yet loaded and construct the arcs $\mathcal{U}(k)$ to represent the feasible changeovers.

Finally, the graph

$$\mathcal{V}(k) = (\mathcal{R} \cup \mathcal{X}(k), \mathcal{U}(k))$$

is formed. The shortest arborescence through the graph $\mathcal{V}(k)$ is then sought. The details of the algorithm used for finding the shortest arborescence are given in appendix B.

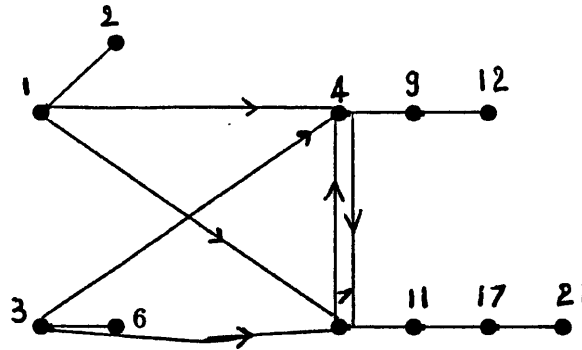
Class of items

The items to be scheduled can be grouped in classes such that the changeover between items of the same class are identical and sequence independent. Moreover, the changeover costs (times) between classes vary only with the class and are sequence dependent.

In the shortest arborescence all the unscheduled items belonging to the same class will be connected to one another, further in the first iteration these items will form a cycle.

In order to avoid unnecessary calculations in the shortest arborescence algorithm, the items belonging to the same class are connected in a chain and the arcs thus created are put in the list of edges. A graph is then created with only the first items in every chain (that is the unconnected item). The problem becomes that of finding the shortest arborescence through the unconnected items. Figure 5.2 shows an example with four classes: (1,2), (3,6), (4,9,12), (7,11,17,21).

Figure 5.2: Connecting items of a same class



5.4.4 Computing penalties on LB

In order to improve the bound obtained in the previous relaxation penalties based on the following proposition are computed.

Proposition 5.1 *Let (j, l) be an arc belonging to the shortest arborescence obtained in solving the relaxed sequencing problem and let*

$$x_{i,j} = 1 \text{ and } x_{i,l} = 1 \text{ in } V(Pr11) \text{ with } i_j \neq i_l$$

then in any optimal solution to $V(Pr)$ one of the following is true:

$$1. \beta_{i,j,l} = 1 \text{ or } \beta_{i_l,j,l} = 1$$

$$2. \beta_{i,j,l} = 1 \text{ if } i \neq i_j \text{ and } i \neq i_l$$

$$3. \beta_{i,j,l} = 0 \text{ } \forall i$$

The proof is straightforward

Proposition 5.2 *Let (j, l) be an arc belonging to the shortest arborescence and i_j and i_l as defined above.*

Consider

$$\bullet ct_{j,l_0} = \min_{l' \neq l} ct_{j,l'} \text{ and } ct_{j_0,l} = \min_{j' \neq j} ct_{j',l}$$

$$\bullet p_{i_l,j} = \min_{i \neq i_j} p_{i,j} \text{ and } p_{i_2,l} = \min_{i \neq i_l} p_{i,l}$$

Define

$$1. p_{n1} = \max(0, p_{i_j,j} - p_{i_l,j})$$

$$2. p_{n2} = \max(0, p_{i_l,l} - p_{i_2,l})$$

$$3. p_{n3} = \max(0, p_{i_j,l} - p_{i_l,l})$$

$$4. p_{n4} = \max(0, p_{i_l,j} - p_{i_j,j})$$

$$5. p_{n5} = \min(ct_{j,l_0}, ct_{j_0,l})$$

similarly define

$$p_n(j, l) = \min(p_{n1}, p_{n2}, p_{n3}, p_{n4}, p_{n5})$$

and

$$PN = \sum_{(j,l)} p_n(j, l) \text{ with } (j, l) \in \mathcal{H}.$$

$$\text{where } \mathcal{H} = \{(j, l) \mid (j, l) \in \mathcal{B} \text{ and } i_j \neq i_l\}$$

with \mathcal{B} the current shortest arborescence. Then

$$LB + PN \leq V(Pr)$$

The proof follows from proposition 5.1.

PN is thus a valid penalty on the LB . To compute PN it is first checked whether the two items forming each arc in the shortest arborescence are loaded onto the same line, if they are not, penalties are computed.

5.5 The overall algorithm for the packing lines

5.5.1 Introduction

As discussed before, to solve problem Pr , an enumerative tree search procedure is adopted. The branching is carried out on the variable $\beta_{i,j,l}$ indicating that item l follows item j on line i . Initially, the problem consists of all item lot sizes and the state of every line at the end of the previous horizon. The tree search starts at the beginning of the current horizon and evolves sequentially in time, setting one item on one line at every node. The relaxation to be resolved at each node deals only with the set of items not yet packed. A depth first procedure was chosen and backtracking occurs when a node is fathomed either by infeasibility or by optimality. The whole procedure can be seen as gradually building a complete schedule for the system.

A desirable property for a relaxation is to have a good chance of providing a feasible solution to the original problem (like for instance an LP, or a Lagrangean relaxation). Unfortunately the relaxation used here does not have this property. Thus, a feasible solution is obtained after all items have been loaded one by one.

The resource constraints are handled separately at every node. Once an item and a line have been selected by the branching rule, the item lot size is split into small sublots corresponding to what the line can pack at each time period. Before loading the sublots onto the line, a check is made, for each time period, to ensure that the resource constraints are satisfied. When they are not, the current time period is left idle and the next period is considered. This is done until all the sublots are loaded or the last time period of the horizon is passed with some sublots yet to be loaded, in which case the node is fathomed.

Since the resource constraints elongate a schedule, when there are ties between items belonging to the same class, the one introducing the minimum idle time is chosen. This is done by carrying out an explicit enumeration. This is possible since the number of items in a given class is small. Alternatively, the item introducing the minimum number of products packed up to the current

time period can be chosen. However, the choice between these two alternatives depends on the problem at hand.

Two feasibility tests are carried out, one is implicit in the solution of the machine loading subproblem and the other is described in the overall algorithm. It is worth stressing the importance of the feasibility tests since if they were not carried out the tree may develop towards an infeasible solution and due to the nature of the depth first selection, a great deal of effort will be spent by the search around a node where an infeasible solution was detected.

Finally, a heuristic solution for providing an initial upper bound to the problem was incorporated and is described in section 5.3.3.

5.5.2 Branching rule

The branching rule is based on priorities and starts by choosing a line, then choosing an item to load onto it.

Step1. Initialisation

Let N be the total number of items to be packed on the horizon, m be the number of items already loaded and T_i be the time left for packing on line i .

Define l_i as the last item set on i and

$$K_j = C_1 \min_i p_{i,j} + C_2 \min_i ct_{l_i,j}$$

$$c_{i,j} = C_1 p_{i,j} + C_2 ct_{l_i,j}$$

For all $j \in \{N - m\}$ find i_j such that

$$c_{i_j,j} = \min_i c_{i,j}$$

with

$$a_{i_j,j} < T_{i_j}$$

Backtracking will occur if there is an item for which there is no such an i_j .

Step2. Selecting the line

For every line i form the set

$$C(i) = \{j \mid (i, j) \in \mathcal{B}\}$$

Where \mathcal{B} is the shortest arborescence at this level of the tree

Choose the line i_b such that

$$|C(i_b)| = \max_i |C_i|$$

Step3. Selecting the item

Define

$$F(j) = K_j - c_{i_b, j'}$$

Among all items such that $i_j = i_b$ choose the one that solves:

$$\max_j F(j)$$

subject to

$$\sum_j a_{i_b, j} x_{i_b, j} + \sum_j \sum_l ct_{j, l} \beta_{i_b, j, l} \leq T$$

at the current node while satisfying the resource constraints.

In case of ties between items of the same class, the one introducing the minimum number of products or the minimum idle time is chosen. Since there is, generally, a small number of items in a class (up to ten), this subproblem is solved using an explicit enumeration procedure.

5.5.3 A heuristic solution

In an attempt to provide an upper bound to the search scheme, a simple heuristic algorithm has been devised. When this solution is close to the optimal, it reduces considerably the size of the tree.

Step1.

As in the branching rule above, and if no i_j is found, the heuristic has failed, stop.

Step2. Priority rule

Among all $j \in \{N - m\}$ find the item j_h which solves:

$$q = \max_j F'(j)$$

where $F'(j) = K_j - c_{i_j, j}$

Set item j_h in line i_{j_h} , where:

$$c_{i_{j_h}, j_h} = \min_i (C_1 p_{i, j} + C_2 c_{l_i, j})$$

where l_i is as defined in the branching rule,

if $q < 0$ then choose the couple (i_h, j_h) such that

$$c_{i_h, j_h} = \min_i \min_j c_{i, j}$$

In both case the load on line i_h should not exceed T_{i_h}

In case of ties between items of the same class resolve as in the step2 of the branching rule.

There is no guarantee that this heuristic will give a feasible solution. Indeed, when the problem is very tight it may fail at step1.

5.5.4 An outline of the algorithm

In this section, an outline of the overall algorithm for the scheduling of the lots on the packing line is presented.

Step0. Initialisation

Compute an upper bound to $V(Pr)$ using the above heuristic,

Initialise $N(s) = \{1, 2, \dots, N\}$, $s = 0$, $T_i = T$ and UB to the value of the heuristic solution, if any, otherwise set $UB = \infty$,

Step1. Lower bounding

Step1a.

Solve problem Pr11 in $N(s)$ with

$$t_i = T_i - PL_i$$

where PL_i is a penalty to account for the changeover, obtaining lb1. If no feasible solution to Pr11 is found then go to step4,

Step1b.

Find the shortest arborescence on the graph consisting of the items in $N(s)$ and the root consisting of items l_i obtaining lb3.

Step1c.

Compute PN the penalty on LB , if $LB + PN \geq UP$ then go to step4,

Step2. Branching step

If $N(s) \neq 0$, perform the steps in the branching rule as described above.

If no couple (i_b, j_b) is found then go to step4,

Feasibility test:

Compute an upper bound to the number of time periods left in the horizon:

$$UT = \sum_t (MAXL - L_t) - B_t$$

where $MAXL$ is the maximum number of lines that can be packing at a given time period, L_t is the number of lines that are packing at time period t and B_t is a lower bound on the changeover times which can be easily derived from the shortest arborescence.

Compute

$$BT = a_{i_b, j_b} + \sum_{j \in N(s)} a_{i_s, j}$$

where i_s corresponds to the optimal loading for j in $Pr11$, if

$$BT > UT$$

go to step4. , otherwise set

$$N(s+1) = N(s) - \{j_b\}$$

$$T_i = T_i - a_{i_b, j_b} - ch_{i, j}$$

$$s = s + 1$$

if

$$\sum_{j \in IS_{i_b}} a_{i_b, j} \leq T_i - PL_i$$

where IS_{i_b} is the set of all items loaded on i_b in the optimal solution to $Pr11$, then update $lb1$ accordingly and go to step1b. , otherwise go to step1a.

If $N(s) = 0$ then

Step3. Feasible solution

Store the solution obtained and update UB

Step4. Backtracking

If $s = 0$ then the incumbent is the optimal solution and stop, otherwise

If at the current node the selected variable $\beta_{i,k,j}$ is set to one, fix it to zero, update $N(s)$ and $T_i(s)$ and go to step1b

otherwise if it is fixed to zero backtrack to the father node: $s=s-1$ go back to step4.

Remarks

1. In order to strengthen the bound lb3, the line selected for branching is the one where the last item has the maximum number of arcs directed from it in the corresponding shortest arborescence. An alternative is to choose the line onto which the maximum number of items was loaded in the machine loading subproblem.
2. Problem $Pr11$ is not solved at every node. When the line and the item chosen at the branching step correspond to the same solution in the previous $Pr11$, lb1 is updated by subtracting p_{i_b,j_b} . This is done even when this is not the case and each unscheduled item can still be loaded on the line on which it was loaded in the solution to the previous machine loading problem. The lower bound is then updated accordingly.
3. At every decision node, in addition to storing the item and the line on which it is packed, the following are also stored:
 The last item that was packed on the line before the current item.
 The time periods where packing starts and ends for the current item.
 The cost of the partial schedule.
4. PL_i depends on the problem at hand and can be determined empirically. In the test problems carried out, PL_i was set to

$$\min_j ch_{i,j}$$

plus as many times the minor changeover as there are items in the class concerned.

5. For every time period it is necessary to keep track of:

The number of lines that are packing.

The total number of material handling systems being used.

The total number of active connections

The number of fitters executing changeovers

The number of active operators.

6. At a backward step, when an item is fathomed, all the free items belonging to the same class could be fathomed without altering optimality. This can be done safely when the problem is not heavily constrained. However, if this is not the case, the search will not be exhaustive. A device has been introduced in the computer implementation, for choosing one or the other option.

7. Finally, when too many lines are dedicated to the packing of a size or a product, the search is not exhaustive since the order in which the lines are selected influences the problem feasibility. To remedy to this situation, a priority between the lines can be included so that a feasible solution is obtained quickly. In such cases, the solution becomes heuristic.

5.5.5 Handling the resource constraints

As mentioned above, in case of tie between items of the same class, the item which introduces either minimum number of products or minimum idle time is selected.

When the number of items in a class is small, this problem can be solved by direct explicit enumeration. In the situation considered here, generally, the number of items of the same format does not exceed ten, therefore an explicit enumeration procedure was implemented. Basically, every item in the class is loaded in turn, checking for every time period if the resource constraints are satisfied. The number of time periods of idle time introduced by the packing of

every item is computed. The item introducing the minimum number of periods of idle time is selected.

Given that the last packing period for the chosen line is t , and that the class s was chosen for this line with

S_p =Set containing the unscheduled items of class (pack size) $p, p = 1, \dots, NC$, the constraints to be satisfied are:

$$\sum_l \left[\frac{n_{l,t}}{S_p} \right] \leq P_t \quad t = t_i, \dots, T_j \quad (5.26)$$

$$m_t \leq NO \quad t = t_i, \dots, T_j \quad (5.27)$$

$$n_{l,t} \leq P I_{l,t} \quad t = t_i, \dots, T_j \quad (5.28)$$

$$\sum_l [n_{l,t}/MCT] \leq MHS \quad t = t_i, \dots, T_j \quad (5.29)$$

Where

NC =Number of pack sizes

MCT =Maximum number of out connections from one material handling system,

m_t =Number of active men at period t ,

$n_{l,t}$ =Number of items of the same base product l , packed at t ,

T_j =Completion period for item j at the current node,

and recalling from chapter four that:

P_t =Maximum number of product that can be packed at time period t ,

$P I_{l,t}$ =Maximum number of items belonging to the same product l that can be packed at time period t

With all parameters as defined in the first section of this chapter and where constraints (5.26) ensure the number of products packed at t does not exceed the maximum allowed, constraints (5.27) ensure that the number of active men

does not exceed the number of men available, constraints (5.28) ensure that the number of items of product l does not exceed the number allowable and constraints (5.29) ensures that the number of active material handling systems does not exceed the total available and $[]$ indicates the upper integer.

5.6 Conclusion

In this chapter, a branch and bound algorithm was developed for the packing lines. The bounds used were based on a relaxation of the linking constraints, that is those constraints linking the sequencing part to the loading part. However, it was suspected that the bounds thus obtained would be low. To improve these bounds, penalties were computed. Though whether there will be improvement or not will certainly depend on the problem data. This feature is common to any branch and bound procedure.

The computational effort depends on the depth of the tree. Near the root, the effort will be function of the problem size; for a large problem, it will be quite substantial. When the tree evolves, the size of the subproblems becomes smaller and the computation effort reduces consequently.

When the current subproblem does not have a feasible solution, it will, generally, take a substantial amount of computing time to prove it, specially near the root. A maximum number of nodes could be fixed for the machine loading subproblem and if there is no feasible solution, the node is fathomed. This is not severe since if the machine loading subproblem is tight, there are less chances of finding a solution when the changeover times are included.

Bearing in mind that the aggregate nature of the long term scheduler will give constrained problems at the lower level, that is problems with few feasible solutions (or even problems with no solution), the search was oriented towards obtaining a feasible solution fairly quickly. If this is successful, the search can be stopped or resumed, depending on what the user thinks of the quality of this solution. When resuming the search, the depth of the backward move can be

made large so as to skip solutions close to the one obtained. Very often, though, the first solution is the optimal and the remainder of the search is used to prove its optimality. It will, then, be sensible to include some criteria for stopping the search. These could be a maximum number of nodes and/or a measure of the gap between the lower bound at the first node and the cost of the solution obtained.

Chapter 6

Short term scheduling for the production-intermediate storage subsystem

6.1 Introduction

The production scheduling of the manufacturing units, with storage limitations, is considered in this chapter. The problem is first formulated as a mixed integer program. However, since a manufacturing unit produces one product at a time and due to the small length of the basic time period, the problem reduces to a (0,1) integer problem. The solution proposed uses a tree search on the integer variables, with bounds computed from the solution of a Lagrangean relaxation of the original problem.

This chapter is organised as follows: in the second section, the subsystem consisting of the manufacturing units and intermediate storage is described and the problem is presented. In the third section, a mixed integer formulation is introduced for the general problem. In the fourth and fifth sections, a solution procedure is developed, for the case where each product has its own storage facility. In the sixth section the solution is extended for the case where the intermediate storage is flexible. In the latter case a model comprising only one

manufacturing unit is considered.

6.2 Description of the problem

6.2.1 Description of the systems

Unlike the packing lines subsystem which is basically the same in the different plants, the manufacturing subsystem varies significantly from one plant to another. However, it consists, invariably, of one or several manufacturing units that process one or more products and a number of intermediate storage units, where products are stored before packing. A manufacturing unit can either be dedicated to the production of one product or general purpose, in that it can produce more than one product, one at a time. Generally, a given product is manufactured at the same rate on all the manufacturing units that produce it.

The intermediate storage consists of silos which, like the manufacturing units, can be either dedicated to one product or flexible. Flexible silos can store different products, one at a time. Sometimes, however, there is no intermediate storage and the products are fed directly to the packing lines. The number of silos, if any, varies from one to ten.

In some plants, the storage facility is only introduced as a buffer between manufacturing units and packing lines; while in others, it is indeed used as a storing facility to allow for long packing runs and packing during periods where production is not possible.

The number of simultaneously active connections between the manufacturing units and the silos (and the packing lines if such connections exist) is limited and the flow of product in the connections is limited. Also, some manufacturing units may not be connected to some silos.

There are two modes of production: in some manufacturing units the production is instantaneous, in others it is delayed. These two terms will be used hereafter to denote the two modes of processing. In a given plant, there could be one or the other type but not both.

- 1. Delayed processing:** In the delayed processing, the production is in discrete amounts, (batches), and it takes the manufacturing unit a certain time to process the batch. There are a lower and a upper limit on the amount of product manufactured by batch, the upper limit being determined by the capacity of the unit, the lower limit by operational consideration, though sometimes processing is carried out at full capacity only.
- 2. Instantaneous processing:** In this case, raw material in the unit reacts instantaneously to form the final product. There is an inflow of raw material and a outflow of final product without delay. There is an upper bound and a lower bound on the rate with which a manufacturing unit produces a given product. Again the amount of production is limited by the maximum capacity of the unit.

In both cases, when changing over from one product to another, a thorough cleaning is needed to avoid cross contamination. In the instantaneous mode, when a unit is set to a new product, it takes a certain time, generally of the order of one hour, to reach the full rate of production. These start up rates are sometimes significantly slow.

The plants operate in one, two or three shifts of eight hours a day. Some decisions are taken at the hourly level particularly in the plants with delayed process units.

6.2.2 The short term commitment

The costs involved in the operation of such systems, in the short term, are mainly changeover costs. However, although the products are manufactured at the same rate on the units where they can be produced, a production pricing can be introduced. This can be advantageous, when for example, one manufacturing unit is dedicated to one product, and another is shared by this product and others. In this case the cost of producing the first product on the second machine can be made higher than in the first.

The changeover costs are introduced by the cleaning process and by the start up rates. Typically start up rates introduce a substantial loss in production. Inventory costs are not considered and there it is only required to use the intermediate storage efficiently, although cleaning cost may be significant in the case of flexible storage. In the case of flexible storage, a cost is allocated to the distribution of silos among the products. For products with high demand, this cost may be such that allocation of silos is encouraged, for those with low demand, high cost will be associated with the allocation of large number of silos.

Any schedule of the manufacturing unit should be consistent with the schedule of the packing lines, in the methodology developed in this thesis, the demand at the production level, for every product, arises from the packing lines schedule. Clearly, having determined the packing lines schedule, the objective is to find an optimal or a good feasible schedule for the manufacturing units which satisfies packing requirements at every time period.

The above problem is remotely related to the problem of lot sizing, although it is much simpler since there are no inventory holding costs involved but rather allocation of a limited storage space in the case of flexible storage. Moreover, in contrast to the multi-product lot sizing problem, at any time period, only one product can be allocated to any unit.

Without the storage space limitation, similar problems (see chapter 3), although more complicated, were studied in particular in [56], where the cost function included set-up cost only, and in [10] where there was a production plus an inventory holding cost to minimise and an ending inventory to build. In both cases the machines were identical. In [55], Love studied a one facility, one product problem with storage limitations where production, set-up and inventory holding cost were to be minimised. He developed a dynamic programming procedure that uses a characterisation of the optimal solution.

6.3 The basic model

In the following, a mixed integer formulation of the problem, which will serve as a basis for further development in the following sections, is introduced. In this model, it is assumed that a product is processed at the same rate in all the machines that manufacture it. Moreover, in the case where storage is available, at any time period a product can be processed on at most one manufacturing unit. This implies that the quantity in storage and the total amount processed by one manufacturing unit at a time period can satisfy the demand at that period. Of course, this is not so when there is no intermediate storage. In this case either demand can be satisfied directly from the manufacturing units without bottlenecks or the problem is infeasible.

Let the manufacturing units be numbered $i = 1, \dots, M$, the products $j = 1, \dots, P$ and the time periods $t = 1, \dots, T$ and define

$y_{i,j,t}$ = amount of product j produced by unit i at t .

$$z_{i,j,t} = \begin{cases} 1 & \text{if unit } i \text{ is processing product } j \text{ at } t \\ 0 & \text{otherwise} \end{cases}$$

$$u_{i,j,t} = \begin{cases} 1 & \text{if unit } i \text{ is set to product } j \text{ at } t \\ 0 & \text{otherwise} \end{cases}$$

$s_{j,t}$ = number of silos allocated to product j at time period t (integer),

$I_{j,t}$ = amount of product j in intermediate storage at the end of t ,

the problem can be defined as

Problem (Ps)

$$\min \sum_i \sum_j \sum_t c_{i,j,t} u_{i,j,t} + r_{i,j,t} z_{i,j,t} + \sum_j \sum_t f_j s_{j,t}$$

subject to

$$\sum_j z_{i,j,t} \leq 1 \quad \forall i, t \quad (6.1)$$

$$\sum_i z_{i,j,t} \leq 1 \quad \forall j, t \quad (6.2)$$

$$u_{i,j,t} \geq z_{i,j,t} - z_{i,j,t-1} \quad \forall i, j, t \quad (6.3)$$

$$y_{i,j,t} \leq X_s z_{i,j,t} \quad \forall i, j, t \quad (6.4)$$

$$I_{j,t} = I_{j,t-1} + \sum_i y_{i,j,t} - d_{j,t} \quad \forall j, t \quad (6.5)$$

$$I_{j,t} \geq d_{j,t+1} \quad \forall j, t \quad (6.6)$$

$$\sum_j s_{j,t} \leq S \quad \forall t \quad (6.7)$$

$$I_{j,t} \leq C_j s_{j,t} \quad \forall j, t \quad (6.8)$$

$$\sum_{t=t_0}^{t_0+mr_j} z_{i,j,t} \geq mr_j z_{i,j,t_0} u_{i,j,t_0} \quad \forall i, j, \quad t_0=1, \dots, T-mr+1 \quad (6.9)$$

$$C_j s_{j,t} \geq I_{j,t-1} \quad \forall j, t \quad (6.10)$$

$$z_{i,j,t} \in \{0, 1\} \quad (6.11)$$

$$s_{j,t} \in \{0, 1, 2\} \quad (6.12)$$

Where

S Number of silos

$d_{j,t}$ is the demand for product j at t

$c_{i,j,t}$ cost of setting j , on i at t

$r_{i,j,t}$ cost of processing product j on i at t

C_j maximum capacity of a silo when filled with product j

mr_j minimum run length for product j in multiples of time periods.

f_j cost of silo allocation for product j

and

- Constraints (6.1) impose that a manufacturing unit processes at most one product at any time period.
- Constraints (6.2) impose that at each time period at most one manufacturing unit is processing a given product
- Constraints (6.3) is the set-up equation : set-up variable is equal to one if manufacturing unit was idle at $t - 1$ and is allocated to producing j at t ,
- Constraints (6.4) ensure that production is allowed only if unit i is allocated to product j at t and is limited by X_s . This bound depends on the storage space available for product j at each time period. An explicit expression for computing X_s at each time period will be given in section 6.4.2.
- Constraints (6.5) are the inventory balance equations
- Constraint (6.6) imposes that the inventory at the end of time period t must be at least equal to demand at time period $t + 1$. This is introduced to cater for the time delay introduced in delayed processing, but they are not valid for instantaneous processing.
- Constraints (6.7) ensure that the number of silos allocated to all products at t must not exceed the total number of silos available

- Constraints (6.8) ensure that the amount in storage at the end of time period t should not exceed the space allocated to product j during period t .
- Constraints (6.9) impose the minimum run length constraint
- Constraints (6.10) ensure storage consistency: storage space allocated during period t should not be less than the amount in storage at the end of period $t - 1$.

The cost function contains the variables $u_{i,j,t}$ and $z_{i,j,t}$. The inclusion of the first variable allows minimisation the number of changeovers whereas the second allows minimisation of the number of time periods allocated product j . If only the first variable was included long run with low production level will be obtained. On the other hand if only the second variable was included small run and more changeover will be obtained. Including both variables will give small, compact run with high production. In order to limit the number of variables, the problem of storage allocation was formulated in terms of number of silos allocated to every product at every period . Indeed, if a schedule silo by silo is considered, the problem will be too large to solve. After obtaining the number of silos for every product, at every time period, the problem of the allocation of silos could be solved using a heuristic.

6.4 Dedicated storage

In a first attempt to solve problem (Ps) , only the case of dedicated storage is considered, in section (6.5), a case with flexible storage and one manufacturing unit is presented. When the storage is dedicated, the last element in the cost function disappears, constraints (6.7) and (6.10) are discarded, the variable $s_{j,t}$ is no longer needed and constraints (6.8) becomes

$$I_{j,t} \leq C_j \quad \forall j, t \quad (6.13)$$

This problem shall be referred to as (Pd) .

6.4.1 The Lagrangean relaxation

The dualisation of constraints (6.1) gives the following Lagrangean problem:

$$(Pdr)_\pi \quad \min \sum_i \sum_j \sum_t c_{i,j,t} u_{i,j,t} + r_{i,j,t} z_{i,j,t} + \sum_i \sum_t \pi_{i,t} (\sum_j z_{i,j,t} - 1)$$

subject to

(6.2), (6.3), (6.4), (6.5), (6.6), (6.9) and (6.13). This set of constraints will be noted U.

When the vector of Lagrange multipliers π is fixed the Lagrangean problem decomposes into P one product subproblems:

$$(Pdr_j)_\pi \quad \min_{z_{i,j,t}, u_{i,j,t} \in U} \sum_i \sum_t c_{i,j,t} u_{i,j,t} + (r_{i,j,t} + \pi_{i,j,t}) z_{i,j,t}$$

These subproblems can be solved by a simple dynamic programming recursion.

6.4.2 The one product subproblem

Due the short length of the basic time period and to the fact that a manufacturing unit processes one product at a time, when unit i is allocated to product j at time period t , the quantity $y_{i,j,t}$ will be computed using a simulator which carries out a simple simulation through the time period. Thus, the one product problem reduces to finding the optimal sequence of $z_{i,j,t}$.

This problem is solved using a dynamic programming procedure. A stage will correspond to a time period t . There will be $M + 2$ states at every stage. Each of the M first stage corresponds to the production of the current product by the relevant unit. The last two stages correspond to the situation where the

current product is not produced at the current time period, in one case with positive amount in intermediate storage while in the other with zero amount. This is of course possible since a product is manufactured at the same rate on the units where it can be manufactured. At the initial stage, the storage corresponds to what it was at the end of the previous horizon.

The ending inventory will be limited only by the space available. This should not introduce major discrepancies since the size of the storage dedicated to a given product depends on the demand for that product. On the other hand, it is clear that when there is no demand for a given product in the current horizon, there will be no production run for it.

The recursion for a one manufacturing unit problem can be represented by the graph shown in figure .1 below, where

- In state (3), the manufacturing unit is producing the current product,
- In state (2), the manufacturing unit is not producing the current product and the amount of product in intermediate storage is positive.
- state (1), is similar to state (2) without product in intermediate storage.

The dynamic programming recursion is

$$F_{st}^j(0) = 0$$

$$I_{j,0} = I_j$$

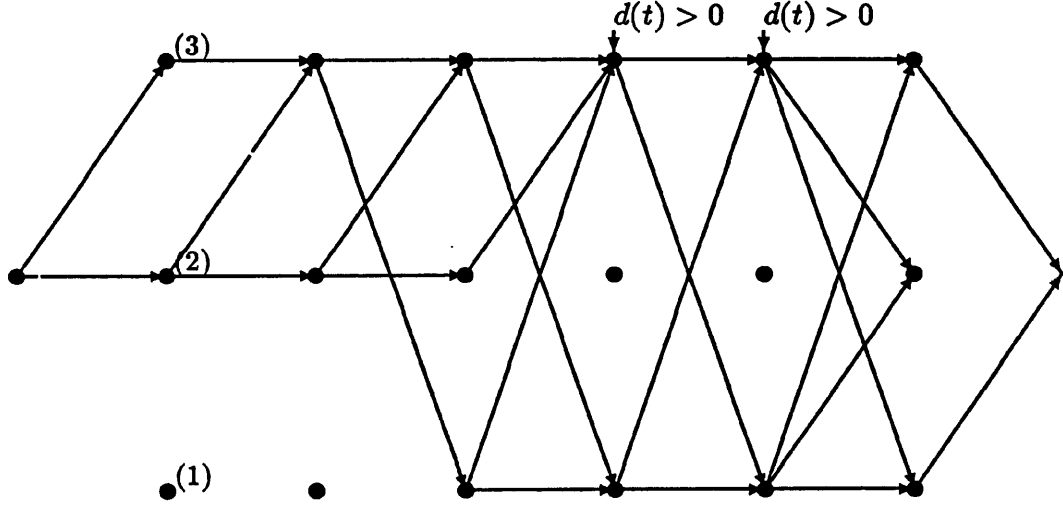
$$F_{st}^j(t) = \min_{st \in U} (F_{st}^j(t-1) + g(st))$$

where I_j is the storage at the end of the previous horizon for product j and

$$g(st) = c_{i,j,t}u_{i,j,t} + (r_{i,j,t} + \pi_{i,j,t})z_{i,j,t}$$

The solution of this problem can be time consuming. However, as will be shown below, the simplicity of the cost function allows the search to be limited to

Figure .1: Network for a one manufacturing unit problem



a few schedules. Consider the one product subproblem when no unit is allocated (in the tree search) to a particular product. Suppose that silo capacity is greater than the minimum run length, allowing for a production run without demand. Due to the form of the cost function, it is clear that if the first period where demand is positive is t_{d1} , then on a given manufacturing unit and prior to t_{d1} the following policies are feasible and have the same cost:

Produce from t_1 , the first period, to $t_1 + mr_j$ and do not from $t_1 + mr_j$ to t_{d1} .

Do not produce at t_1 , produce from t_2 to $t_2 + mr_j$ and do not from $t_2 + mr_j$ to t_{d1} .

And in general, do not produce prior to t_k , produce from t_k to $t_k + mr_j$ and do not from $t_k + mr_j$ to t_{d1} where $k = 1, \dots, t_{d1} - mr_j$ and $t_{d1} - mr_j \geq 1$.

There is no loss of optimality in discarding all policies that allow for production prior to $t_{d1} - mr_j$. Moreover, among all the above policies, the policy that starts production at $t_{d1} - mr_j$ minimises the waiting time in intermediate storage.

Therefore, instead of considering all the time periods in the horizon for the

one product subproblem, the recursion is started at $\max(t_1, t_d - mr_j)$. The recursions are then carried out normally until period t_e , where there is no demand and the minimum run length constraint is satisfied. All periods from t_e to $t_{d2} - mr_j$, t_{d2} the next period with positive demand, can be skipped, the state of the units remaining stationary from t_e until $t_{d2} - mr_j$. This procedure is repeated until there is no more demand.

In the case where any of the periods t_{di} , with positive demand is allocated to a particular product, in the tree search, the beginning of the recursion is shifted backward. This is done until a feasible solution is found or the initial time period or the last period of the previous production run has been reached without finding a feasible recursion, in which case there is no feasible solution.

The simulator

As mentioned above, due to the structure of the system and to the mode of processing, a simulator which computes the quantity to produce is employed. Basically the simulation through the time period takes into account the mode of process, the storage space available at the beginning of the time period, the demand and the flows.

At any time period, the policy will be to produce as much as the storage space allows. This is possible and should not lead to infeasibility due to unresolvable bottlenecks since the length of the basic time period is small enough, of the order of four hours.

Ideally, the demand is taken out of storage at a steady flow and production is made at constant rate during the whole time step. However, this is not the case here. Indeed, in the schedule for the packing liness, packing was allowed during a time period where minor changeovers are executed. This means that demand for product to pack is not uniform during the time period. Nevertheless, this constraint is ignored and it is assumed that whenever there is demand, it is uniformly distributed (of course, to take this into account a smaller time period must be considered, thus increasing the size of the problem).

On the other hand, due to the start up times, production rates are not constant, at least during the first hours or so of manufacturing a given product. During these start up times, the production rate is supposed to evolve linearly in time until it reaches its nominal value.

Two simulators are introduced, one for each mode of processing.

Instantaneous processing In this case, the following is supposed:

1. The rate of production evolves linearly during the start up time
2. The start up time is less than the length of the time period
3. demand is either taken from storage or directly from the manufacturing units.

The maximum amount of product j that manufacturing unit i can produce in a given time period is

$$y_n = ((t_{i,j}^s v_{i,j} + (H - t_{i,j}^s) r n_{i,j})) u_{i,j,t} + r n_{i,j} H (1 - u_{i,j,t}) z_{i,j,t}$$

where

- $t_{i,j}^s$ =start up time of unit i for product j
- $v_{i,j}$ =speed of unit i in reaching nominal rate of product j
- $r n_{i,j}$ =nominal rate of production of unit i when producing product j
- H =length of a time period in hours

Given that the storage available at the beginning of time period t is $C_j - I_{j,t-1}$ if the demand is taken into account, the upper bound on the production of j at t is

$$X_s = C_j - (I_{j,t-1} - d_{j,t})$$

Thus the amount of product j produced by unit i is

$$y_{i,j,t} = \min(X_s, y_n)$$

A lower bound can be imposed on $y_{i,j,t}$.

Delayed processing In this case the following is assumed:

1. There are no start up rates.
2. At the end of a time period, a unit has to be empty.
3. There is a minimum production by batch.
4. The length of a time period is greater than the sum of the time to process full batch and the time to empty the unit.
5. If the unit cannot be emptied during the time period then production has to be decreased.
6. A unit can be connected to only one silo at a time.

Let

- $y_{i,max}$ =Maximum capacity of unit i ,
- f_{max} =maximum flow out of a unit to storage,
- Y_b =amount produced in batch b ,
- t_b =time to empty batch b ,
- t_{rb} = time to produce batch b ,
- y_{min} =minimum production allowable by batch,
- Y_{binc} = increment of production,
- X_s as defined above.

The simulation is carried out as follows

0. $t_p = 0$

1. $t_b = 0, Y_b = 0$

2. Fix the batch size to the maximum possible: either the full capacity of unit j or the maximum space available

$$Y_b = \min(y_{i,max}, X_s)$$

If $Y_b \leq y_{min}$, stop production for this time period.

otherwise

3. compute the time to empty unit

1) If the batch size is greater than the space available empty at maximum outflow i.e.

$$Y_b > X_s \text{ then } t_b = Y_b / f_{max}$$

2) If the batch is smaller and there is a demand, empty, at maximum flow until there is no more space then adjust flow out of unit to flow out of storage i.e.

$$t_b = X_s / f_{max} + (Y_b - X_s) / d_{j,t}$$

3) If there is no space, adjust flow out of unit to flow out of storage i.e.

$$t_b = Y_b / d_{j,t}$$

4. Time elapsed is $t_p = t_p + t_{rb} + t_b$

If $(H - t_p) > 0$ then total produced in period t so far is:

$$y_{i,j,t} = y_{i,j,t} + Y_b$$

and the amount in storage is:

$$I_{j,t} = I_{j,t-1} - d_{j,t} + Y_b + I_{j,t}$$

$$b = b + 1 \text{ go to 1,}$$

if $(H - t_p) < 0$, then

$$b = b,$$

$$t_p = t_p - t_{rb} - t_b$$

$$Y_b = Y_b - Y_{binc} \text{ go to 3.}$$

6.5 The solution procedure

6.5.1 Overview

In order to solve problem (Pd) , a binary depth first tree search procedure on the variables $z_{i,j,t}$, corresponding to the decision to allocate manufacturing unit i to product j at time period t , has been adopted. At a given node of the tree, a number of variables is set to zero or one and the remainder is free. An attempt at fathoming the node is made by computing the Lagrangean dual. Fathoming occurs when the lower bound obtained is greater than the incumbent and when the subproblem has no feasible solution. When fathoming is successful, a backward step is carried out; when it is not, a forward step is carried by selecting the next variable to set to one or zero, according to the following branching rule.

6.5.2 Branching rule

At first the couple (i_p, t_p) , i.e. machine and time period, for which

$$\pi_{i,t} \sum_j (z_{i,j,t} - 1)$$

is maximum, is selected.

After constructing the set

$$J_p = \{j \mid z_{i_p,j,t_p} = 1\}$$

the product $j_p \in J_p$ for which

$$\min_{t_j} (t_j - t_p)$$

is selected where

t_j is such that $d_{j,t_j} > 0$

with

$$t_j > t_p, \quad d_{j,t} = 0 \quad \forall t \mid t_p < t < t_j \quad j \in J_p$$

In other words, the most overloaded line, for all time periods, is selected and the product for which the next demand, after the time period where the overload has occurred, is the closer is selected.

Then set

$$z_{i_p, j_p, t_p} = 1$$

$$z_{i_p, j, t_p} = 0 \quad \forall j \in J_p, \quad j \neq j_p$$

and store the subproblems

$$z_{i_p, j_p, t_p} = 0$$

$$z_{i_p, j, j_p} = 1 \quad \forall j \in J_p, \quad j \neq j_p$$

in the list of subproblems.

6.5.3 The first node

At the first node of the tree, the relaxation of problem (Prd), with all variables free, is considered, the vector of Lagrange multipliers is initialised to zero, and a large number (>15) of iterations of the subgradient algorithm is performed. To update the Lagrange multipliers, the following formula is used,

$$\pi_{i,t}^{k+1} = \pi_{i,t}^k + \gamma (\sum_j z_{i,j,t} - 1) (V(Pd)^* - V(Pdr)_\pi) / \sum_{i,t} (\sum_j z_{i,j,t} - 1)^2$$

with $V(Pd)^*$ being an approximation to $V(Pd)$

$0 < \gamma \leq 2$ γ is initialised to 2 and halved when the lagrangean fails to increase after 5 iterations.

and $V(Pdr)_\pi$ is the solution to the Lagrangean problem with the current multipliers.

If, at any iteration, the solution to the current Lagrangean problem satisfies the complementary slackness conditions, that is

$$\pi_{i,t}(\sum_j z_{i,j,t} - 1) = 0 \quad \forall i, t$$

it is optimal and feasible in the original problem and the procedure stops. If, on the other hand, it happens (which is very unlikely here) that the solution is feasible but does not satisfy the complementary slackness that is:

$$\sum_j (z_{i,j,t} - 1) \leq 0 \quad \forall i, t$$

the solution is stored and the search may be stopped. When the maximum number of iterations is reached, the best $V(Pdr)_\pi$, the corresponding multipliers are stored and the tree search is started.

6.5.4 General node

First, one variable is set to one while others are set to zero according to the branching rule. A lower bound is then computed for the subproblem thus obtained, by performing a small number of iterations of the subgradient algorithm. The Lagrange multipliers and γ are initialised to the best obtained at the father node and updated as in the first node. If at any iteration, the lower bound is greater than the best solution so far or any of the one product subproblem is infeasible, the node is fathomed. The node is also fathomed when the solution satisfies the complementary slackness, in which case it is stored as incumbent. On the other hand, if the solution is feasible but does not satisfy the complementary slackness, it is also stored as incumbent. In this latter case and in case where no feasible solution is obtained and the node is not fathomed a forward step is carried out.

6.5.5 Backtracking

The last subproblem stored in the list is selected, the Lagrange multipliers are set to the best multipliers obtained at the first node, the number of iterations is fixed to an average number (>8). If the variable defining the current subproblem is set to one, it is fixed to zero and another ,free, variable, corresponding to loading an alternative product on the same line, at the same time period is set to one, creating a brother node. If no such variable can be found, all the variables that were fixed at this level are freed and backtracking to the father node occurs. If the list is empty the procedure is stopped. The incumbent corresponds to the optimal solution.

6.5.6 Updating parameters of the coordination process

The updating of the parameters $P(t)$ and $PI(j,t)$ of the coordination device, discussed in chapter 4, can now be presented. Recalling the following:

$P(t)$: maximum number of products that can be packed simultaneously in the whole packing room at period t ,

$PI(j,t)$: maximum number of items belonging to the product j that can be packed during time period t ,

$PA(t)$: number of products packed at time period t in the current packing lines schedule,

$PIA(j,t)$: number of items of product j , packed during t in the optimal packing line schedule,

and that when at Step 4. of the coordination algorithm no feasible solution is obtained at the manufacturing units level, a new pass r has to be carried out. For this $P'(t)$ and $PI'(j,t)$ are to be updated so that a new packing lines solution satisfying them is sought. When there is no feasible solution to the manufacturing units subproblem two situations may occur: either the search stopped prematurely because a one product subproblem is infeasible or the search ended without finding a feasible solution. For each of these two cases,

the updating will be different:

1) The search stopped at time period t_b for product j . In this case, $PI^r(j_t, t_b)$ is updated as follows:

$$PI^r(j_t, t_b) = PIA(j_t, t_b) - pid$$

where pid is fixed by the user.

In other words, the maximum number of items belonging to product j that can be packed at time period t_b is decreased in order to reduce the demand for this product at that time period.

2) The search ended without finding a feasible schedule. In this case, t_b can be chosen as one of the time periods where branching was carried out in the manufacturing units subproblem. Here, the vector $P(t_b)$ is updated as follows:

$$P^r(t_b) = PA(t_b) - pd$$

where pd is fixed by the user. The maximum number of products that can be packed in the packing room at time period t_b is decreased in an attempt to alleviate the bottleneck provoked by product competing for the same machine at time period t_b . This procedure is of course applicable whatever the mode of production or the silos type, flexible or dedicated.

6.6 A case with flexible storage

6.6.1 Extension of the basic model

In this section, a problem with flexible storage is considered. For reasons of dimensionality, a case with only one manufacturing unit is considered. More manufacturing units can, of course, be considered by requiring that the variable $s_{j,t}$ takes only two values, thus limiting the sizing of the dynamic programming procedure. To keep the generality, the variable $z_{i,j,t}$ will be left with the subscript i .

The mathematical model given in the third section is again taken as basis. In order to simplify the silos allocation, total production during the time horizon will be forced to equal the cumulative requirement, thus imposing that ending inventory for all products to be zero:

$$I_{j,T} = 0 \quad \forall j \quad (6.14)$$

In order to avoid allocation of silos that will stay unused during the whole time period, a supplementary constraint is introduced. A convenient way of expressing this constraint is to impose that

if for any t, j

$$s_{j,t} - [I_{j,t}/C_j] > 0$$

and

$$s_{j,t} - [I_{j,t-1}/C_j] > 0$$

then the corresponding state is infeasible. These constraints will be referred to as 6.16.

These constraints ensures that a state is infeasible if the number of empty silos is positive at its beginning and at its end.

6.6.2 The Lagrangean relaxation

As in (Pdr) , a search tree with Lagrangean relaxation is developed. However, in addition to dualising constraints (6.1) with multipliers $\pi_{i,t}$ constraints (6.7) are also dualised with multipliers ξ_t , obtaining the new Lagrangean problem:

Problem $(Psr)_{(\pi, \xi)}$

$$\min \sum_j \sum_t c_{1,j,t} u_{1,j,t} + (r_{1,j,t} + \pi_{1,j,t}) z_{1,j,t} + \sum_j \sum_t f_j s_{j,t} + \sum_t \xi_t (\sum_j s_{j,t} - S)$$

Subject to (6.3), (6.4), (6.5), (6.6), (6.7), (6.8), (6.9), (6.10), (6.11), (6.12) (6.14) and (6.16). These constraints will be referred to as CR . Again,

the Lagrangean problem decomposes into P subproblems when the multipliers π and ξ are fixed, giving:

$$(Psr_j)_{(\pi, \xi)} \quad \min_{z_{i,j,t}, u_{i,j,t}, s_{j,t} \in CR} \sum_t c_{1,j,t} + (r_{1,j,t} + \pi_{1,j,t})z_{1,j,t} + (\xi_t + f_j)s_{j,t}$$

6.6.3 The solution procedure

As in the dedicated storage case, the quantity to produce in one time period is computed using the simulator.

The mode of processing in this case is instantaneous, the simulator for the instantaneous mode is used. The formula for computing the space available is altered to take into account the variable $s_{j,t}$. Thus:

$$X_s = C_j s_{j,t} - (I_{j,t-1} - I_{j,t})$$

On the other hand, since the ending inventory is forced to be zero for all products the equation for computing the variable $y_{i,j,t}$ becomes:

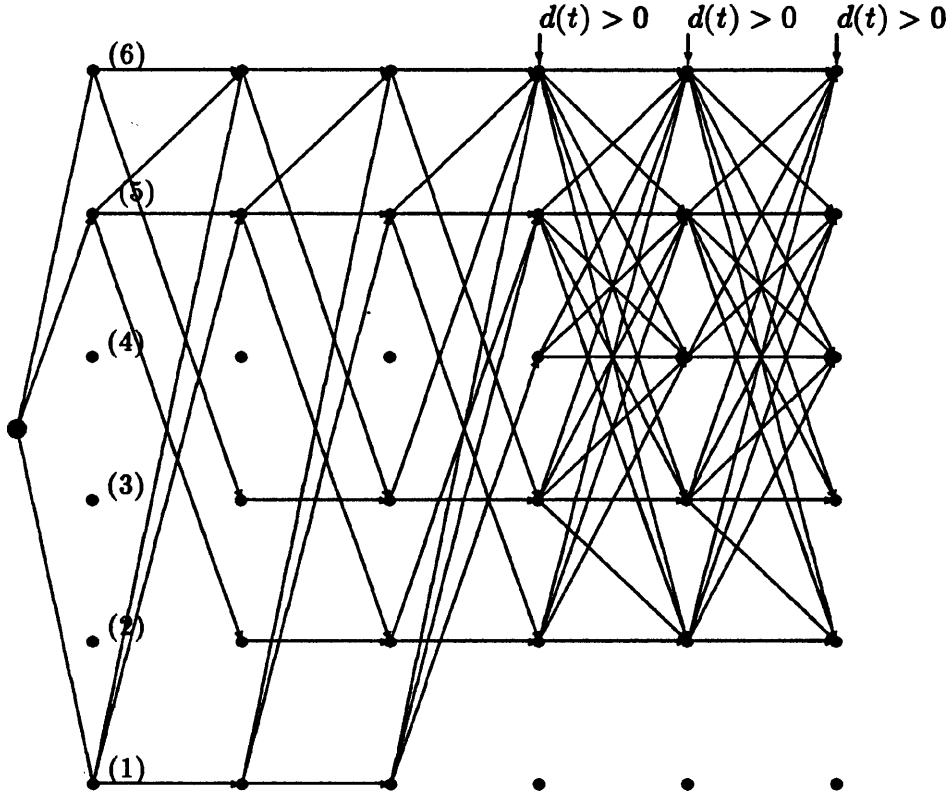
$$y_{i,j,t} = \min \{X_s, y_n, \sum_{t=t}^{t=T} d_{j,t}\}$$

Each of the resulting P integer subproblems is solved using a dynamic programming procedure. In this case, each state corresponds to one of the two states (on or off) of the manufacturing unit together with the number of silos allocated to the product. As mentioned above the number of silos can be zero, one or two. This gives the acyclic graph shown in figure 6.2, where:

- In state(6) the manufacturing unit is producing the current product and two silos allocated to this product.
- state(5) as state(6) with one silo
- state(4) as state (6) but no silo.
- In state(3) the manufacturing unit is idle and two silos are allocated to the current product.

- state (2) as state (3) with one silo
- state (1) as state (3) with no silo.

Figure 6.2: Network for the flexible storage problem



The considerations introduced for the dedicated storage case are still valid here. The dynamic programming recursion will be the same as above except that

$$F_{st}^j = \min_{st \in CR} (F_{st}^j(t-1) + g'(st))$$

$$g'(st) = g(st) + (f_j + \xi_t)s_{j,t}$$

The solution procedure is the same as before but the branching rule is slightly altered to account for the silos allocation.

6.6.4 Branching rule

As long as the solution of the relaxed problem does not satisfy (6.1) , the same branching rule as in problem *Pd* is invoked. When these are satisfied and constraints on the number of silos, (6.7), are not, branching is again carried out on the variable $z_{1,j,t}$ but according to the following rule

1. Find the first t for which $\sum s_{j,t} - S > 0$, call this $t(s)$

2. Let $J_s = \{j \mid s_{j,t_m} > 0\}$ then

For all $j \in J_s$ find the products with a run greater than the minimum run length, prior to t_m , if there are none, fathom the node, otherwise,

3. choose the product with the longest run, say j_g , and the last period of this run, t_g , and set

$$z_{1,j_g,t_g} = 0$$

The motivation behind the above rule is that instead of trying to reallocate the silos for the time period where the constraint was violated, the longest run is shortened, hoping that this will force feasibility by reducing the storage needed by the product concerned. This is done until either feasibility is obtained or no run is longer than the minimum run length while the storage space limit is violated, in which case the node is fathomed.

6.7 Conclusion

In this chapter, a solution methodology was proposed for the production-storage stage. A tree search procedure based on a Lagrangean relaxation was adopted. The small length of the basic time period and the fact that a manufacturing

unit can produce at most one product at a time period allowed the real valued variable, here the amount to produce in one time period, to be calculated using a simple simulation procedure. The minimum run length constraint was formulated in terms of multiple of time periods. However for problems where the length of the time step is relatively large, this can no longer be possible. It can be argued that the minimum run length constraint is not important since it is sufficient to minimise the number of set-up to satisfy it. When this is not so, there are no feasible solutions that satisfy the minimum run length and the packing line schedule must be perturbed.

The simplicity of the cost function allowed a reduction of the effort needed for solving the one product subproblems by considering only some particular schedules.

For the flexible silos case, the silos allocation was formulated in terms of number of silos allocated to a particular product in each time period. The silo allocation was priced so that high demand product would be allocated more space than low demand products. It is possible not to include the silos allocation explicitly in the cost function, rather in the dynamic programming procedure priority will be given to one or another value of $s_{j,t}$. In this way, silos allocation can be seen as heuristic. This version was actually implemented.

Here as in the packing lines subsystem, due to the tightness of the problem, a feasible solution is often sufficient.

Chapter 7

Results

7.1 Introduction

In this chapter, some experiments carried out on a number of test problems are described.

In the first section, computational results involving only the packing lines are presented. In particular, the behaviour of the algorithm under the variations of structural and operational constraints is analysed. The data were generated randomly.

In the second section, two-stage problems are considered. The data for these problems were gathered from real life manufacturing facilities. The lot sizes are output from a long term scheduler.

Full data for all problems are given in appendix C for the problems considered in the first section and in appendix D for those considered in the second section. The program was written in FORTRAN77 and run on a PRIME 9955.

7.2 Packing lines algorithm

It is not easy to generate data to test the algorithm, particularly when it comes to testing a particular problem for feasibility. Indeed, there is a risk that the problem generated may be too easy, in this particular case, a problem with

too slack a demand. In this respect, for each structure, demand was gradually increased until it was impossible to obtain a feasible solution after 3000 nodes.

The problems tested here have a number of characteristics in common. These characteristics are based on a real life situation. The pack sizes are divided into two or three groups: small, large and medium (if any). Similarly, the packing lines are divided into two or three subgroups, according to the sizes they can pack. In particular, the lines that pack a particular group cannot pack any item of any other group. An item can be packed on all the lines of the group to which it belongs, with different rates. The best packing cost is allocated to the line corresponding to the best packing rate. Change over cost between items sharing the same best line are low whereas higher changeover costs are incurred between items which do not share the same best line.

The horizon is of ten shifts of eight hours each, the length of the basic time period is always one hour. Results are reported for four basic structures, a five lines problem ($P5$), a six lines problem ($P6$) and two seven lines problem ($P71$) and ($P72$). The behaviour of the algorithm is analysed under variation of the demand. Also, the effects of the distribution of items between the products for problem ($P5$) and the effect of varying the operational and structural constraints for the other problems are analysed.

7.2.1 Problem ($P5$)

In this structure, the seven silos are dedicated and feed directly the packing lines. A silo can feed one packing line at a time, thus at each time period only one item of each product can be packed. There are seven products, twelve sizes and twenty three items.

Results are shown in table 7.1. Three size/product distributions were tested. In the first, ($P5_1$), the number of sizes per product does not vary too much with the products; in the two other ($P5_2$), ($P5_3$), some products have a high number of items while others have just one. Two demands were analysed; one small, $D1$, and the other large, $D2$. $P5_1$ was tested with both $D1$ ($P5_{10}$) and $D2$

($P5_{11}$). ($P5_2$) and ($P5_3$) were tested with $D1$, ($P5_{20}$) and ($P5_{30}$). These last two problems were also run with the possibility of packing two items of the same products at a time period, ($P5_{21}$) and ($P5_{31}$). As can be seen in the table although problem ($P5_1$) was relatively easy to solve for demand $D1$, the search did not terminate for problems ($P5_2$) and ($P5_3$) with the same demand. This is due to the fact that for these product distributions the lower bounds are too far from the optimum and thus are no longer of any help.

7.2.2 Problem (P6)

In this problem, the silos are flexible and feed the packing lines through six materiel handling systems. There are seven products, fourteen sizes and twenty five items.

Results are shown in table 7.2 Five demand distribution were tested. The optimal solution for problems with slack demand ($P6_4$) and ($P6_5$) was found early in the search. This was not the case for problems with higher demand. In particular, for problem ($P6_3$) the algorithm did not find a better solution than the heuristic. This latter problem was rerun with a higher level of manpower. In this case, two solutions were found and the search ended after 2945 nodes and 65s. The lower bound at the first node was the same in all problems.

7.2.3 Problems ($P71$) and ($P72$)

In problem ($P71$) there are seven packing lines fed by three material handling systems. There are eight products, fifteen sizes and thirty items.

Results are shown in table 7.3. Again low and high demand were tested and the same features as above were observed.

All problems were rerun with a precision of 5% within the optimum, i.e. fathoming was allowed if the lower bound was greater or equal to 95% of the incumbent. In this case, problems ($P71_1$) and ($P71_2$) ended after 57 nodes and 97 nodes respectively while problem ($P71_3$) did not end after 3275 nodes. The

lower bound at the first node is slightly higher for problems ($P71_1$) and ($P71_2$) than for the other two problems.

Problem ($P72$) consists of seven packing lines fed by four material handling systems. There are ten products, sixteen sizes and thirty items. Globally, the same results (table 7.4) as before were obtained.

The following abbreviations are used in the tables:

- *HS*: heuristic solution,
- *LB1*: lower bound at the first node,
- *Nsol*: Number of solution obtained,
- *Fsol*: first solution obtained,
- *Lsol*: last solution obtained,
- *Opt*: Optimality: if Yes, the optimum was found, if No it was not,
- *CPU time*: total CPU time elapsed before either the optimum was found or the search was stopped,
- *Nodes*: total number of nodes generated,
- *Node*: node at which the solution was obtained,
- *Cost*: cost of the solution.

7.3 Computational results for two-stage systems

In this section, some experimental results concerning the detailed scheduling of production for two production-storage-packing facilities are presented. The two facilities correspond to two plants operated by a major producer of detergents. The two plants studied were chosen because they are typical of many others and

Table 7.1: Results for problems series *P5*

<i>Problem</i>	<i>HS</i>	<i>LB1</i>	<i>Nsol</i>	<i>Fsol</i>		<i>Lsol</i>		<i>Opt</i>	<i>CPU Time</i>	<i>Nodes</i>
				<i>Node</i>	<i>Cost</i>	<i>Node</i>	<i>Cost</i>			
<i>P5₁₀</i>		52	1	24	60			<i>Yes</i>	8s	310
<i>P5₁₁</i>		52	2	28	63	1986	62	<i>No</i>	90s	3000
<i>P5₂₀</i>		52	1	306	72			<i>No</i>	80s	3200
<i>P5₂₁</i>	62	52	1	24	61			<i>No</i>	60s	3000
<i>P5₃₀</i>		52	1	100	79			<i>No</i>	70s	3000
<i>P5₃₁</i>	62	52	1	24	61			<i>No</i>	60s	3000

Table 7.2: Results for problems series *P6*

<i>Problem</i>	<i>HS</i>	<i>LB1</i>	<i>Nsol</i>	<i>Fsol</i>		<i>Lsol</i>		<i>Opt</i>	<i>CPU Time</i>	<i>Nodes</i>
				<i>Node</i>	<i>Cost</i>	<i>Node</i>	<i>Cost</i>			
<i>P6₁</i>	80	66	1	26	79			<i>No</i>	50s	2400
<i>P6₂</i>		66	1	170	72			<i>Yes</i>	32s	1545
<i>P6₃</i>	81	66	0					<i>No</i>		3000
<i>P6₄</i>	75	66	1	106	71			<i>Yes</i>	12s	461
<i>P6₅</i>	74	66	2	26	71	194	69	<i>Yes</i>	10s	385

Table 7.3: Results for problems series *P71*

<i>Problem</i>	<i>HS</i>	<i>LB1</i>	<i>Nsol</i>	<i>Fsol</i>		<i>Lsol</i>		<i>Opt</i>	<i>CPU Time</i>	<i>Nodes</i>
				<i>Node</i>	<i>Cost</i>	<i>Node</i>	<i>Cost</i>			
<i>P71₁</i>		82	2	31	112	87	103	<i>No</i>	120s	3000
<i>P71₂</i>		82	2	31	114	124	106	<i>No</i>	80s	1000
<i>P71₃</i>		80	5	65	123	1347	110	<i>No</i>	100s	3280
<i>P71₄</i>		80	2	619	116	681	101	<i>Yes</i>	25s	790

Table 7.4: Results problems series *P72*

<i>Problem</i>	<i>HS</i>	<i>LB1</i>	<i>Nsol</i>	<i>Fsol</i>		<i>Lsol</i>		<i>Opt</i>	<i>CPU Time</i>	<i>Nodes</i>
				<i>Node</i>	<i>Cost</i>	<i>Node</i>	<i>Cost</i>			
<i>P72₁</i>	84	79	3	31	83	331	81	<i>Yes</i>	12s	409
<i>P72₂</i>	81	97						<i>Yes</i>	11s	24
<i>P72₃</i>		79	2	51	100	743	99	<i>Yes</i>	13s	847
<i>P72₄</i>		79	2	43	108	49	102	<i>No</i>	90s	3500
<i>P72₅</i>		81	2	43	119	173	102	<i>No</i>	122s	3600

are, structure-wise and operations-wise, very dissimilar. The first plant will be referred to as *PSP1*, the second as *PSP2*.

Although the lot sizes are output from the long term scheduler, due to the aggregate nature of the model at this level, there is no guarantee that an overall feasible solution exists. Therefore,

it will be necessary, sometimes, to alter the lot sizes in order to obtain feasibility. For this reason, in the following, overall feasibility, rather than optimality, will be analysed. Coordination was carried out by the user. In order to be acquainted with each system, a number of small problems, not reported here, were tested. In all test problems, only the first packing lines solution was considered. In the charts giving the packing lines schedules, the first number indicates the product, while the second indicates the pack size.

7.3.1 Computational results for *PSP1*

Full data for this problem are presented in appendix D, here only the main characteristics are presented. The facility consists of:

- a) One manufacturing unit of the continuous type operated three shifts a day. It manufactures four products.
- b) Four small capacity silos, each dedicated to one product and acting mainly as buffers.
- c) Three high capacity, general purpose silos.
- e) Two material handling system. This implies that only two products can be packed simultaneously.
- d) Six packing lines operated two shifts a day. Packing rates are relatively high, compared to the manufacturing unit rate. Changeover times are not sequence dependent. Major changeovers are of eight hours each and minor changeover times are of one hour each. There is a pool of thirty five operators.

The long term horizon consists of four weeks, the short term of one week, that is ten shifts of eight hours. Four tests (*T11*, *T12*, *T13*, *T14*) were carried out on a rolling horizon basis, each corresponding to a basic time period of the

long term horizon. For all the tests, the length of the basic time period for the manufacturing unit was four hours whereas that for the packing lines was one hour. . Since every day the manufacturing unit operates one shift more than the packing lines, the last shift for the manufacturing unit of the preceding horizon was allocated to building initial storage for the current horizon. Due to low rates at the manufacturing unit, the maximum quantity that can be packed of a given product at any time period (hour) was fixed to 40 tons. The minimum run length constraint was expressed in terms of the minimum quantity to produce at the time period where the set up was carried out. This quantity was set to 20 tons for all four problems.

Test T11

The initial packing lines schedule is shown in figure 7.1, and the resulting demand for base products is shown in Table A1. This demand did not give a feasible schedule. Infeasibility occurred for product 1 at time period 4, which correspond to time periods twelve to sixteen at the packing lines level. At these time periods four items of product 1 were packed. The maximum number of items packed for this product at time period 16 was decreased from four to three. On the other hand product 3 was delayed, ie packing of any item of this product was forbidden from period 27 to 36, because there is likely to be a bottleneck at these periods due to heavy demand for product two. The new packing lines schedule is not shown, again it did not give a feasible schedule. Infeasibility this time occurred for product 3 at time period 16. The maximum number of items of this product that can be packed was decreased from 4 to 3 for hour 40 to 47. Two more iterations were carried out before infeasibility occurred at the packing lines (no solution was found after 2000 nodes). The last packing lines schedule is shown in figure 7.2 and the corresponding product demand in table A2. The altered demand is shown in table A3. The production schedule and the silos allocation are shown in tables 7.5 and 7.6 Initial inventories were: 110 tons for product 1 and 68 tons for product 4. Feasibility was achieved at

100% for products 1 and 2, at 97.4% for product 3 and at 93% for product four.

Test T12

The demand (table B1) arising from the initial packing lines schedule (7.3) gave no solutions. This demand is too scattered, in order to "lump" it, packing of product 2 was forbidden from period one to period 16 and that of product four was forbidden from period 1 to 48. This gave the packing lines schedule shown in figure 7.4 and the demand shown in table B2. There was no feasible schedule at the manufacturing unit but this was due to the minimum run length constraints: there were small isolated demands for products 1 and 2 at periods 13 and 19 that could not be satisfied from storage. Eliminating these two negligible lots gave the demand shown in table B3. The production schedule and silos allocation are shown in tables 7.3 and 7.4 Initial inventories were: 117 tons for product 1 and 68 tons for product 3. Feasibility was achieved at: 98% for product 1 , 98% for product 2 and 100% for product 3 and 4.

Test T13

Again, the initial packing lines schedule (7.5) did not give any solution at the manufacturing unit level. Here, too, the demand (table C1) is too scattered, and in order to lump it, packing of product 3 was forbidden from time period 1 to 34 and that of product 1 from 20 to 30. Also, only packing of one item of product 1 was allowed from period 33 to 45 since it seems that there will be bottleneck at these periods due to heavy demand for product 3. Two other iterations were carried out in order to resolve bottleneck involving product 1 and 3. Before that however the schedule was frozen, ie packing of product 2 was forbidden from period 36 to 80 and that of product 4 from 34 to 80. The last schedule is shown in figure 7.6 and demand in table C2. This demand was altered as in table C3. Initial inventories were 72 tons for product 1 and 68 tons for product 4. Feasibility was achieved at 94% for product 1, at 100% for product 2, at 96% for product 3 and at 93% for product 4.

Test T_{14}

The strategy employed is as before: try to lump the product demand: this was done mainly for product 3. Then bottleneck involving product 1 and 2 at periods 1 to 4 was resolved by decreasing the number of items packed for product 1 to one from period 1 to 8 and to zero from period 8 to 16. The final packing lines schedule is shown in figure 7.8 and the products demand in table D1. The altered demand, production schedule and silos allocation are shown in tables D3, 7.9 and 7.10 respectively. Initial inventories were: 118 for product 1 and 72 for product 2. Feasibility was achieved at 99% for product 1, and at 100% for products 2, 3 and 4.

7.3.2 Computational results for $PSP2$

As for problem $PSP1$ complete data are in appendix. The plant consists of: a) Four manufacturing units of the batch type: unit 1 is dedicated to product 1 unit 2 makes product 1,2,3,4,5; unit 3 is dedicated to product 6 and unit 4 is dedicated to product 7.

b) Seven silos, all dedicated to a particular product. There are no material handling systems. At any time there may be at most one connection from one silo to the packing lines, thus simultaneous packing of items of the same product is not possible.

c) Four packing lines. Packing line three is dedicated to packing product 6 and 7. Changeovers are not sequence dependent. Major changeover times are of eight hours, minor changeover times were less than one hour and not taken into account.

There are eight weeks in the long term horizon and every short term is made of two weeks. The length of the basic time period for the packing lines subproblems is of two hours and that of the manufacturing units is of four hours. Again, only the first solution at packing lines was considered and as before four test problems (T_{21} , T_{22} , T_{23} , T_{24}) corresponding to time periods of the long

term horizon. were carried out. For every horizon, initial storage was taken as the ending storage at the previous horizon.

Test T21

The initial packing lines schedule and the corresponding demand are shown in figure 7.9 and table E1. No feasible solution was obtained. The following perturbations were introduced:

1. Do not pack product 1 from period 1 to 6.
2. Do not pack product 2 from 1 to 4.
3. Do not pack product 3 from 1 to 20.
4. Do not pack product 5 from 1 to 50.

The reminder of the perturbations was to introduce delays, particularly for product 1 which production rate is relatively low. Feasibility was achieved at 100% for all products. Initial storage of 10 tons was introduced for product 7.

Tests T22, T23 and T24

For all these tests, perturbations concerned mainly product 1, and were carried out in order to introduce delays in the packing of this product. Feasibility was achieved at 100% for all products. All results are shown in the corresponding tables and figures. Initial storage were the accumulated storage from previous horizon.

Abbreviations used in the tables

Pr: Product

Tp: time period

Pi: Product number i ,

Amt: Amount produced of the current product during the current time period

Mi Manufacturing unit i

When Pr, the product number, is equal to zero, the corresponding manufacturing unit is idle. In all tables showing the demand arising from the packing lines schedule, the first column indicates the time period. In the second column, the number 1 indicates that the packing line can pack in the corresponding time period, ** indicates that it cannot pack. All the other columns correspond to the demand for the products, in tons (column 3: demand for product 1, column 4: demand for product 2, etc...).

Figure 7.1: T11: First schedule for the packing lines

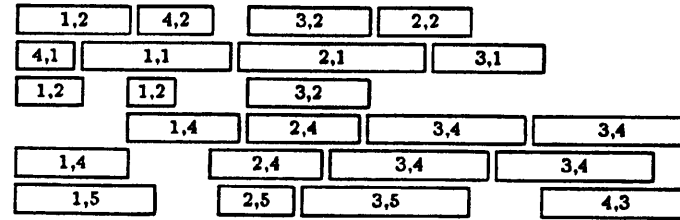


Table A1 T11: Demand corresponding to the first packing lines schedule

1	1	113.4000	0.0000	0.0000	6.0000	0.0000	0.0000	0.0000
2	1	151.2000	0.0000	0.0000	6.0000	0.0000	0.0000	0.0000
3	1	131.3600	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
4	1	131.3600	0.0000	0.0000	14.8000	0.0000	0.0000	0.0000
5	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
7	1	69.1600	0.0000	0.0000	29.6000	0.0000	0.0000	0.0000
8	1	55.7600	38.1000	0.0000	14.8000	0.0000	0.0000	0.0000
9	1	12.2000	132.4000	24.8000	0.0000	0.0000	0.0000	0.0000
10	1	0.0000	146.9000	49.6000	0.0000	0.0000	0.0000	0.0000
11	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
12	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
13	1	0.0000	83.4000	105.6000	0.0000	0.0000	0.0000	0.0000
14	1	0.0000	32.6000	133.0000	0.0000	0.0000	0.0000	0.0000
15	1	0.0000	35.2000	134.4000	0.0000	0.0000	0.0000	0.0000
16	1	0.0000	29.8000	137.6000	0.0000	0.0000	0.0000	0.0000
17	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
18	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
19	1	0.0000	14.0000	118.4000	0.0000	0.0000	0.0000	0.0000
20	1	0.0000	0.0000	84.8000	0.0000	0.0000	0.0000	0.0000
21	1	0.0000	0.0000	81.6000	18.6000	0.0000	0.0000	0.0000
22	1	0.0000	0.0000	89.6000	37.2000	0.0000	0.0000	0.0000
23	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
24	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
25	1	0.0000	0.0000	89.6000	37.2000	0.0000	0.0000	0.0000
26	1	0.0000	0.0000	67.2000	37.2000	0.0000	0.0000	0.0000
27	1	0.0000	0.0000	22.4000	18.6000	0.0000	0.0000	0.0000
28	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
29	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
30	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Figure 7.2: *T11*: Last schedule for the packing lines

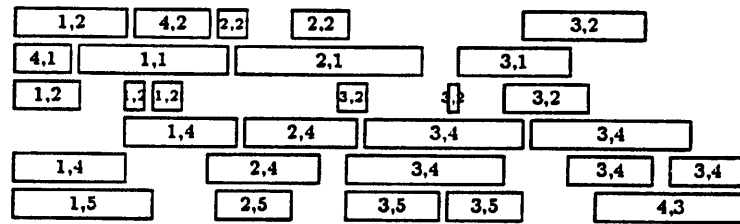


Table A2 *T11*: Demand corresponding to the last packing lines schedule

1	1	112.4000	0.0000	0.0000	6.0000	0.0000	0.0000	0.0000
2	1	151.2000	0.0000	0.0000	6.0000	0.0000	0.0000	0.0000
3	1	131.3600	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
4	1	124.6600	0.0000	0.0000	14.8000	0.0000	0.0000	0.0000
5	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
7	1	75.8600	0.0000	0.0000	29.6000	0.0000	0.0000	0.0000
8	1	55.7600	45.1000	0.0000	14.8000	0.0000	0.0000	0.0000
9	1	12.2000	146.4000	0.0000	0.0000	0.0000	0.0000	0.0000
10	1	0.0000	153.9000	0.0000	0.0000	0.0000	0.0000	0.0000
11	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
12	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
13	1	0.0000	111.4000	0.0000	0.0000	0.0000	0.0000	0.0000
14	1	0.0000	46.6000	97.0000	0.0000	0.0000	0.0000	0.0000
15	1	0.0000	7.2000	134.4000	0.0000	0.0000	0.0000	0.0000
16	1	0.0000	1.8000	123.2000	0.0000	0.0000	0.0000	0.0000
17	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
18	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
19	1	0.0000	0.0000	145.4000	0.0000	0.0000	0.0000	0.0000
20	1	0.0000	0.0000	139.8000	0.0000	0.0000	0.0000	0.0000
21	1	0.0000	0.0000	89.6000	0.0000	0.0000	0.0000	0.0000
22	1	0.0000	0.0000	123.4000	0.0000	0.0000	0.0000	0.0000
23	**	0.0000	0.0000	0.0000	0.0000	0.0050	0.0000	0.0000
24	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
25	1	0.0000	0.0000	114.4000	37.2000	0.0000	0.0000	0.0000
26	1	0.0000	0.0000	73.4000	37.2000	0.0000	0.0000	0.0000
27	1	0.0000	0.0000	67.2000	37.2000	0.0000	0.0000	0.0000
28	1	0.0000	0.0000	44.8000	37.2000	0.0000	0.0000	0.0000
29	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
30	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table A3 T11: Altered demand

1	1	113.4000	0.0000	0.0000	6.0000	0.0000	0.0000	0.0000
2	1	151.2000	0.0000	0.0000	6.0000	0.0000	0.0000	0.0000
3	1	131.3600	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
4	1	124.6600	0.0000	0.0000	14.0000	0.0000	0.0000	0.0000
5	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
7	1	75.8600	0.0000	0.0000	28.0000	0.0000	0.0000	0.0000
8	1	55.7600	45.1000	0.0000	14.0000	0.0000	0.0000	0.0000
9	1	12.2000	146.4000	0.0000	0.0000	0.0000	0.0000	0.0000
10	1	0.0000	153.9000	0.0000	0.0000	0.0000	0.0000	0.0000
11	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
12	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
13	1	0.0000	111.4000	0.0000	0.0000	0.0000	0.0000	0.0000
14	1	0.0000	46.6000	97.0000	0.0000	0.0000	0.0000	0.0000
15	1	0.0000	7.2000	130.0000	0.0000	0.0000	0.0000	0.0000
16	1	0.0000	1.8000	98.0000	0.0000	0.0000	0.0000	0.0000
17	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
18	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
19	1	0.0000	0.0000	145.4000	0.0000	0.0000	0.0000	0.0000
20	1	0.0000	0.0000	130.8000	0.0000	0.0000	0.0000	0.0000
21	1	0.0000	0.0000	89.6000	0.0000	0.0000	0.0000	0.0000
22	1	0.0000	0.0000	123.4000	0.0000	0.0000	0.0000	0.0000
23	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
24	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
25	1	0.0000	0.0000	114.4000	34.0000	0.0000	0.0000	0.0000
26	1	0.0000	0.0000	73.4000	34.0000	0.0000	0.0000	0.0000
27	1	0.0000	0.0000	67.2000	34.0000	0.0000	0.0000	0.0000
28	1	0.0000	0.0000	44.8000	34.0000	0.0000	0.0000	0.0000
29	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
30	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 7.1: *T11*: Production schedule

Tp	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
Pr	1	1	1	1	1	2	1	2	2	2	2	3	2	3	3	3	3	3	3	3	3	3	4	3	3	3	4	3	0	0	
Amt	104	104	104	104	66.62	72	71	72	104	104	104	68	56	46	89	96	96	99	46	59	96	96	96	68	68	96	82	56	84	0	0

Table 7.2: *T11*: Silos allocation

Pr	Number of silos per time period																														
P1	2	2	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
P2	0	0	0	0	0	1	1	2	2	1	2	2	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	2	2	1	1	1	0	2	2	1	1	0	0	0	0
P4	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0	0	0

Figure 7.3: T12: First schedule for the packing lines

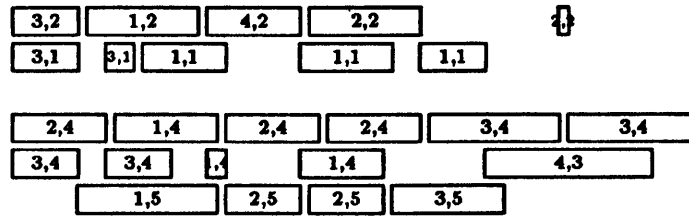


Table B1 T12: Demand corresponding to the first packing lines schedule

1	1	0.0000	38.1000	57.0000	0.0000	0.0000	0.0000	0.0000
2	1	0.0000	50.8000	76.0000	0.0000	0.0000	0.0000	0.0000
3	1	68.9000	38.1000	12.8000	0.0000	0.0000	0.0000	0.0000
4	1	126.1400	0.0000	48.0000	0.0000	0.0000	0.0000	0.0000
5	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
7	1	131.3600	0.0000	22.4000	0.0000	0.0000	0.0000	0.0000
8	1	111.2600	0.0000	0.0000	14.8000	0.0000	0.0000	0.0000
9	1	0.0000	101.6000	0.0000	29.6000	0.0000	0.0000	0.0000
10	1	0.0000	101.6000	0.0000	29.6000	0.0000	0.0000	0.0000
11	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
12	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
13	1	55.7600	97.2000	0.0000	0.0000	0.0000	0.0000	0.0000
14	1	55.7600	129.6000	0.0000	0.0000	0.0000	0.0000	0.0000
15	1	15.6800	91.5000	22.4000	0.0000	0.0000	0.0000	0.0000
16	1	5.2200	19.7000	67.2000	0.0000	0.0000	0.0000	0.0000
17	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
18	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
19	1	6.9600	0.0000	89.6000	0.0000	0.0000	0.0000	0.0000
20	1	0.0000	0.0000	67.2000	37.2000	0.0000	0.0000	0.0000
21	1	0.0000	0.0000	44.8000	37.2000	0.0000	0.0000	0.0000
22	1	0.0000	7.0000	33.6000	37.2000	0.0000	0.0000	0.0000
23	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
24	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
25	1	0.0000	0.0000	44.8000	37.2000	0.0000	0.0000	0.0000
26	1	0.0000	0.0000	44.8000	18.6000	0.0000	0.0000	0.0000
27	1	0.0000	0.0000	33.6000	0.0000	0.0000	0.0000	0.0000
28	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
29	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
30	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Figure 7.4: T12: Last schedule for the packing lines

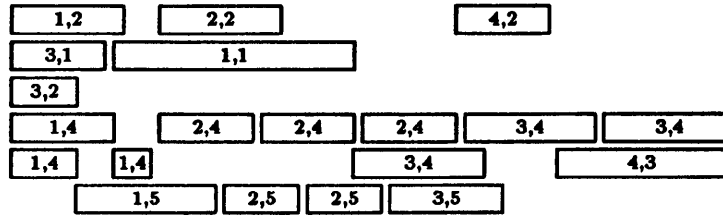


Table B2 T12: Demand corresponding to the last packing lines schedule

1	1	93.3000	0.0000	23.4000	0.0000	0.0000	0.0000	0.0000
2	1	124.4000	0.0000	31.2000	0.0000	0.0000	0.0000	0.0000
3	1	124.4000	0.0000	4.8000	0.0000	0.0000	0.0000	0.0000
4	1	111.2600	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
5	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
7	1	55.7600	59.1000	0.0000	0.0000	0.0000	0.0000	0.0000
8	1	43.5600	78.8000	0.0000	0.0000	0.0000	0.0000	0.0000
9	1	6.9600	116.9000	0.0000	0.0000	0.0000	0.0000	0.0000
10	1	6.9600	115.6000	0.0000	0.0000	0.0000	0.0000	0.0000
11	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
12	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
13	1	6.9600	88.9000	0.0000	0.0000	0.0000	0.0000	0.0000
14	1	3.4800	88.9000	22.4000	0.0000	0.0000	0.0000	0.0000
15	1	0.0000	63.5000	67.2000	0.0000	0.0000	0.0000	0.0000
16	1	0.0000	50.8000	89.6000	0.0000	0.0000	0.0000	0.0000
17	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
18	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
19	1	0.0000	12.7000	112.0000	22.2000	0.0000	0.0000	0.0000
20	1	0.0000	0.0000	67.2000	29.6000	0.0000	0.0000	0.0000
21	1	0.0000	0.0000	44.8000	22.2000	0.0000	0.0000	0.0000
22	1	0.0000	0.0000	44.8000	37.2000	0.0000	0.0000	0.0000
23	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
24	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
25	1	0.0000	0.0000	33.6000	37.2000	0.0000	0.0000	0.0000
26	1	0.0000	0.0000	44.8000	37.2000	0.0000	0.0000	0.0000
27	1	0.0000	0.0000	44.8000	37.2000	0.0000	0.0000	0.0000
28	1	0.0000	0.0000	33.6000	18.6000	0.0000	0.0000	0.0000
29	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
30	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table B3 T12: Altered demand

1	1	93.3000	0.0000	23.4000	0.0000	0.0000	0.0000	0.0000
2	1	124.4000	0.0000	31.2000	0.0000	0.0000	0.0000	0.0000
3	1	124.4000	0.0000	4.8000	0.0000	0.0000	0.0000	0.0000
4	1	111.2600	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
5	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
7	1	55.7600	59.1000	0.0000	0.0000	0.0000	0.0000	0.0000
8	1	43.5600	78.8000	0.0000	0.0000	0.0000	0.0000	0.0000
9	1	6.9600	116.9000	0.0000	0.0000	0.0000	0.0000	0.0000
10	1	6.9600	115.6000	0.0000	0.0000	0.0000	0.0000	0.0000
11	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
12	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
13	1	0.0000	88.9000	0.0000	0.0000	0.0000	0.0000	0.0000
14	1	0.0000	88.9000	22.4000	0.0000	0.0000	0.0000	0.0000
15	1	0.0000	63.5000	67.2000	0.0000	0.0000	0.0000	0.0000
16	1	0.0000	50.8000	89.6000	0.0000	0.0000	0.0000	0.0000
17	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
18	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
19	1	0.0000	0.0000	112.0000	22.2000	0.0000	0.0000	0.0000
20	1	0.0000	0.0000	67.2000	29.6000	0.0000	0.0000	0.0000
21	1	0.0000	0.0000	44.8000	22.2000	0.0000	0.0000	0.0000
22	1	0.0000	0.0000	44.8000	37.2000	0.0000	0.0000	0.0000
23	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
24	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
25	1	0.0000	0.0000	33.6000	37.2000	0.0000	0.0000	0.0000
26	1	0.0000	0.0000	44.8000	37.2000	0.0000	0.0000	0.0000
27	1	0.0000	0.0000	44.8000	37.2000	0.0000	0.0000	0.0000
28	1	0.0000	0.0000	33.6000	18.6000	0.0000	0.0000	0.0000
29	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
30	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 7.3: T12: Production schedule

Tp	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Pr	1	1	1	1	1	3	2	2	2	2	2	2	2	2	3	3	4	4	3	3	3	3	3	3	4	4	3	0	0	0
Amt	94.8	104.1	104.1	104.1	42.8	63.4	72.1	104.1	104.1	104.1	58.4	46.5	58.9	84.7	68.9	68.4	75.8	79.4	44.8	44.8	46.5	46.5	55.4	71.3	38.3	0.	0.	0.	0.	0.

Table 7.4: T12: Silos allocation

P	Number of silos per time period																														
P1	2	2	2	2	2	2	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
P2	0	0	0	0	0	0	0	1	1	0	2	2	2	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P3	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	2	2	2	1	1	0	0	
P4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	2	2	1	1	0	0	0	1	1	0	0	0	

Figure 7.5: T13: First schedule for the packing lines

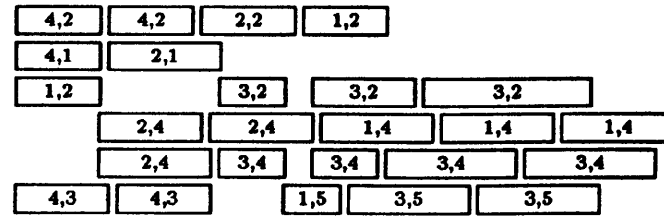


Table C1 T13: Demand corresponding to the first packing lines schedule

1	1	20.1000	0.0000	0.0000	56.1000	0.0000	0.0000	0.0000
2	1	26.8000	0.0000	0.0000	74.8000	0.0000	0.0000	0.0000
3	1	13.4000	52.6000	0.0000	54.1000	0.0000	0.0000	0.0000
4	1	0.0000	108.8000	0.0000	66.8000	0.0000	0.0000	0.0000
5	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
7	1	0.0000	108.8000	0.0000	66.8000	0.0000	0.0000	0.0000
8	1	0.0000	89.9000	17.4000	18.6000	0.0000	0.0000	0.0000
9	1	0.0000	78.8000	69.6000	0.0000	0.0000	0.0000	0.0000
10	1	24.4000	71.8000	34.8000	0.0000	0.0000	0.0000	0.0000
11	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
12	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
13	1	100.0000	12.7000	52.2000	0.0000	0.0000	0.0000	0.0000
14	1	75.6000	0.0000	103.2000	0.0000	0.0000	0.0000	0.0000
15	1	55.5000	0.0000	103.2000	0.0000	0.0000	0.0000	0.0000
16	1	36.6000	0.0000	108.2000	0.0000	0.0000	0.0000	0.0000
17	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
18	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
19	1	48.8000	0.0000	103.2000	0.0000	0.0000	0.0000	0.0000
20	1	48.8000	0.0000	103.2000	0.0000	0.0000	0.0000	0.0000
21	1	36.6000	0.0000	114.4000	0.0000	0.0000	0.0000	0.0000
22	1	48.8000	0.0000	108.2000	0.0000	0.0000	0.0000	0.0000
23	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
24	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
25	1	48.8000	0.0000	44.8000	0.0000	0.0000	0.0000	0.0000
26	1	48.8000	0.0000	22.4000	0.0000	0.0000	0.0000	0.0000
27	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
28	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
29	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
30	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Figure 7.6: T13: Last schedule for the packing lines

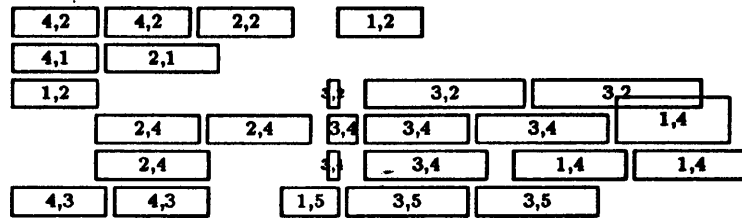


Table C2 T13: Demand corresponding to the last packing lines schedule

1	1	20.100	0.000	0.000	56.100	0.000	0.000	0.000
2	1	26.800	0.000	0.000	74.800	0.000	0.000	0.000
3	1	13.400	52.600	0.000	54.100	0.000	0.000	0.000
4	1	0.000	108.800	0.000	66.800	0.000	0.000	0.000
5	**	0.000	0.000	0.000	0.000	0.000	0.000	0.000
6	**	0.000	0.000	0.000	0.000	0.000	0.000	0.000
7	1	0.000	108.800	0.000	66.800	0.000	0.000	0.000
8	1	0.000	89.900	0.000	18.600	0.000	0.000	0.000
9	1	0.000	78.800	0.000	0.000	0.000	0.000	0.000
10	1	24.400	71.800	0.000	0.000	0.000	0.000	0.000
11	**	0.000	0.000	0.000	0.000	0.000	0.000	0.000
12	**	0.000	0.000	0.000	0.000	0.000	0.000	0.000
13	1	48.800	12.700	28.600	0.000	0.000	0.000	0.000
14	1	26.800	0.000	84.600	0.000	0.000	0.000	0.000
15	1	26.800	0.000	159.200	0.000	0.000	0.000	0.000
16	1	6.700	0.000	159.200	0.000	0.000	0.000	0.000
17	**	0.000	0.000	0.000	0.000	0.000	0.000	0.000
18	**	0.000	0.000	0.000	0.000	0.000	0.000	0.000
19	1	0.000	0.000	136.800	0.000	0.000	0.000	0.000
20	1	12.200	0.000	114.400	0.000	0.000	0.000	0.000
21	1	48.800	0.000	108.200	0.000	0.000	0.000	0.000
22	1	48.800	0.000	114.400	0.000	0.000	0.000	0.000
23	**	0.000	0.000	0.000	0.000	0.000	0.000	0.000
24	**	0.000	0.000	0.000	0.000	0.000	0.000	0.000
25	1	61.000	0.000	36.000	0.000	0.000	0.000	0.000
26	1	97.600	0.000	24.800	0.000	0.000	0.000	0.000
27	1	97.600	0.000	18.600	0.000	0.000	0.000	0.000
28	1	73.200	0.000	0.000	0.000	0.000	0.000	0.000
29	**	0.000	0.000	0.000	0.000	0.000	0.000	0.000
30	**	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table C3 T13: Altered demand

1	1	20.1000	0.0000	0.0000	56.1000	0.0000	0.0000	0.0000
2	1	26.8000	0.0000	0.0000	74.8000	0.0000	0.0000	0.0000
3	1	13.4000	52.6000	0.0000	54.1000	0.0000	0.0000	0.0000
4	1	0.0000	108.8000	0.0000	60.5000	0.0000	0.0000	0.0000
5	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
7	1	0.0000	108.8000	0.0000	68.0000	0.0000	0.0000	0.0000
8	1	0.0000	89.9000	0.0000	0.0000	0.0000	0.0000	0.0000
9	1	0.0000	78.8000	0.0000	0.0000	0.0000	0.0000	0.0000
10	1	24.4000	84.0000	0.0000	0.0000	0.0000	0.0000	0.0000
11	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
12	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
13	1	48.8000	0.0000	28.6000	0.0000	0.0000	0.0000	0.0000
14	1	26.8000	0.0000	84.6000	0.0000	0.0000	0.0000	0.0000
15	1	26.8000	0.0000	130.0000	0.0000	0.0000	0.0000	0.0000
16	1	6.7000	0.0000	130.0000	0.0000	0.0000	0.0000	0.0000
17	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
18	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
19	1	0.0000	0.0000	136.8000	0.0000	0.0000	0.0000	0.0000
20	1	12.0000	0.0000	114.4000	0.0000	0.0000	0.0000	0.0000
21	1	30.0000	0.0000	108.2000	0.0000	0.0000	0.0000	0.0000
22	1	30.0000	0.0000	96.0000	0.0000	0.0000	0.0000	0.0000
23	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
24	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
25	1	61.0000	0.0000	36.0000	0.0000	0.0000	0.0000	0.0000
26	1	97.6000	0.0000	24.8000	0.0000	0.0000	0.0000	0.0000
27	1	97.6000	0.0000	18.6000	0.0000	0.0000	0.0000	0.0000
28	1	73.2000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
29	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
30	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 7.5: T13: Production schedule

Tp	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Pr	4	4	2	2	2	4	2	2	2	1	1	0	3	3	3	3	3	1	3	3	3	3	3	0	1	1	1	1	0	0
Amt	96.81	97.21	104.10	103.96	97.67	104.67	72.59	20.68	96.96	96.96	96.96	62.66	88.96	96.96	96.96	75.07	72.10	41.04	49.40	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 7.6: T13: Silos allocation

P	Number of silos per time period																													
P1	1	1	0	0	0	0	0	0	0	1	2	2	2	1	1	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0
P2	0	0	0	0	2	2	2	2	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	1	2	2	1	2	2	2	1	0	0	2	2	2	1	0	0	0	0
P4	2	2	2	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7.7: T14: First schedule for the packing lines

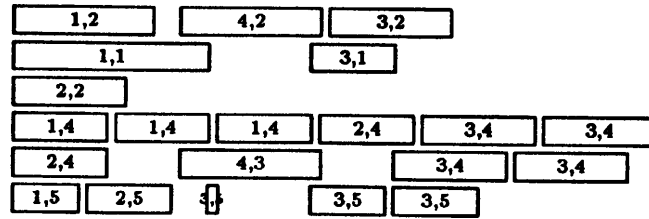


Table D1 T14: Demand corresponding to the first packing lines schedule

1	1	98.5200	59.1000	0.0000	0.0000	0.0000	0.0000	0.0000
2	1	131.3600	78.8000	0.0000	0.0000	0.0000	0.0000	0.0000
3	1	70.3600	104.2000	0.0000	0.0000	0.0000	0.0000	0.0000
4	1	82.5600	57.8000	0.0000	0.0000	0.0000	0.0000	0.0000
5	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
7	1	55.7600	25.4000	0.0000	16.7000	0.0000	0.0000	0.0000
8	1	40.0800	0.0000	11.2000	66.8000	0.0000	0.0000	0.0000
9	1	48.8000	0.0000	0.0000	66.8000	0.0000	0.0000	0.0000
10	1	48.8000	0.0000	0.0000	66.8000	0.0000	0.0000	0.0000
11	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
12	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
13	1	12.2000	25.4000	45.8000	33.4000	0.0000	0.0000	0.0000
14	1	0.0000	50.8000	77.6000	0.0000	0.0000	0.0000	0.0000
15	1	0.0000	50.8000	84.8000	0.0000	0.0000	0.0000	0.0000
16	1	0.0000	0.0000	149.0000	0.0000	0.0000	0.0000	0.0000
17	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
18	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
19	1	0.0000	0.0000	123.2000	0.0000	0.0000	0.0000	0.0000
20	1	0.0000	0.0000	78.4000	0.0000	0.0000	0.0000	0.0000
21	1	0.0000	0.0000	78.4000	0.0000	0.0000	0.0000	0.0000
22	1	0.0000	0.0000	89.6000	0.0000	0.0000	0.0000	0.0000
23	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
24	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
25	1	0.0000	0.0000	78.4000	0.0000	0.0000	0.0000	0.0000
26	1	0.0000	0.0000	22.4000	0.0000	0.0000	0.0000	0.0000
27	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
28	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
29	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
30	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Figure 7.8: : T14: Last schedule for the packing lines

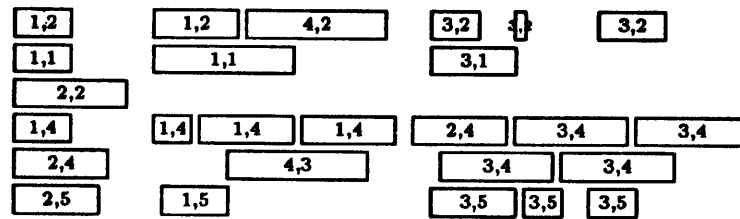


Table D1 T14: Demand corresponding to the last packing lines schedule

1	1	61.9200	97.2000	0.0000	0.0000	0.0000	0.0000	0.0000
2	1	61.9200	129.6000	0.0000	0.0000	0.0000	0.0000	0.0000
3	1	0.0000	91.5000	0.0000	0.0000	0.0000	0.0000	0.0000
4	1	0.0000	7.0000	0.0000	0.0000	0.0000	0.0000	0.0000
5	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
7	1	119.1600	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
8	1	119.1600	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
9	1	62.4600	0.0000	0.0000	52.0000	0.0000	0.0000	0.0000
10	1	41.8200	0.0000	0.0000	66.8000	0.0000	0.0000	0.0000
11	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
12	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
13	1	48.8000	0.0000	0.0000	66.8000	0.0000	0.0000	0.0000
14	1	48.8000	0.0000	0.0000	57.5000	0.0000	0.0000	0.0000
15	1	24.4000	0.0000	0.0000	7.4000	0.0000	0.0000	0.0000
16	1	0.0000	50.8000	50.0000	0.0000	0.0000	0.0000	0.0000
17	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
18	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
19	1	0.0000	50.8000	116.2000	0.0000	0.0000	0.0000	0.0000
20	1	0.0000	25.4000	101.8000	0.0000	0.0000	0.0000	0.0000
21	1	0.0000	0.0000	123.2000	0.0000	0.0000	0.0000	0.0000
22	1	0.0000	0.0000	100.8000	0.0000	0.0000	0.0000	0.0000
23	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
24	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
25	1	0.0000	0.0000	148.0000	0.0000	0.0000	0.0000	0.0000
26	1	0.0000	0.0000	108.2000	0.0000	0.0000	0.0000	0.0000
27	1	0.0000	0.0000	44.8000	0.0000	0.0000	0.0000	0.0000
28	1	0.0000	0.0000	44.8000	0.0000	0.0000	0.0000	0.0000
29	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
30	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table D3 T14:Altered demand

1	1	59.0000	97.2000	0.0000	0.0000	0.0000	0.0000	0.0000
2	1	59.0000	129.6000	0.0000	0.0000	0.0000	0.0000	0.0000
3	1	0.0000	91.5000	0.0000	0.0000	0.0000	0.0000	0.0000
4	1	0.0000	7.0000	0.0000	0.0000	0.0000	0.0000	0.0000
5	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
6	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
7	1	119.1600	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
8	1	119.1600	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
9	1	62.4600	0.0000	0.0000	52.0000	0.0000	0.0000	0.0000
10	1	41.8200	0.0000	0.0000	66.8000	0.0000	0.0000	0.0000
11	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
12	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
13	1	48.8000	0.0000	0.0000	66.8000	0.0000	0.0000	0.0000
14	1	48.8000	0.0000	0.0000	57.5000	0.0000	0.0000	0.0000
15	1	24.4000	0.0000	0.0000	7.4000	0.0000	0.0000	0.0000
16	1	0.0000	50.8000	50.0000	0.0000	0.0000	0.0000	0.0000
17	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
18	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
19	1	0.0000	50.8000	116.2000	0.0000	0.0000	0.0000	0.0000
20	1	0.0000	25.4000	101.8000	0.0000	0.0000	0.0000	0.0000
21	1	0.0000	0.0000	123.2000	0.0000	0.0000	0.0000	0.0000
22	1	0.0000	0.0000	100.8000	0.0000	0.0000	0.0000	0.0000
23	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
24	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
25	1	0.0000	0.0000	148.0000	0.0000	0.0000	0.0000	0.0000
26	1	0.0000	0.0000	108.2000	0.0000	0.0000	0.0000	0.0000
27	.1	0.0000	0.0000	44.8000	0.0000	0.0000	0.0000	0.0000
28	1	0.0000	0.0000	44.8000	0.0000	0.0000	0.0000	0.0000
29	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
30	**	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 7.9: T14: Production schedule

Tp	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
Pr	2	2	2	1	1	4	1	1	1	4	4	4	4	1	3	2	2	3	3	3	3	3	3	3	3	3	3	0	0	0
Amt	72.1	104.9	5.8	72.4	6.5	68.7	2.1	104.1	104.6	51.3	46.5	16.7	66.1	68.7	236.5	54.9	66.9	66.9	66.9	66.9	66.9	66.9	66.9	66.9	66.9	66.9	66.9	66.9	66.9	66.9

Table 7.10 T14: Silos allocation

P	Number of silos per time period																													
P1	2	1	0	2	2	2	1	2	2	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
P2	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	2	2	2	0	0	0	0	0	0	0	0	0	0	0	0
P3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0	2	2	2	1	1	1	0	0
P4	0	0	0	0	0	1	1	1	1	0	2	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 7.9: T21: First schedule for the packing lines

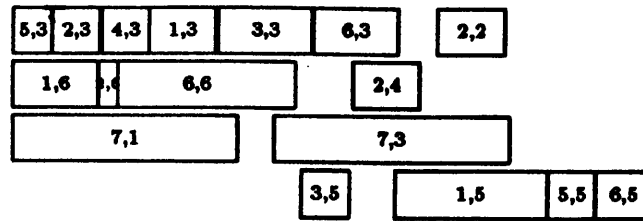


Table E1 T21: Demand corresponding to the first packing lines schedule

1	1	15.0000	0.0000	0.0000	0.0000	11.2500	0.0000	4.5000
2	1	20.0000	0.0000	0.0000	0.0000	15.0000	0.0000	6.0000
3	1	20.0000	11.2500	0.0000	0.0000	0.0000	0.0000	6.0000
4	1	20.0000	15.0000	0.0000	0.0000	0.0000	0.0000	6.0000
5	1	10.0000	12.5000	0.0000	3.7500	0.0000	0.0000	6.0000
6	1	0.0000	10.0000	0.0000	15.0000	0.0000	5.0000	6.0000
7	1	0.0000	0.0000	0.0000	15.0000	0.0000	20.0000	6.0000
8	1	11.2500	0.0000	0.0000	0.0000	0.0000	20.0000	6.0000
9	1	15.0000	0.0000	0.0000	0.0000	0.0000	20.0000	6.0000
10	1	15.0000	0.0000	0.0000	0.0000	0.0000	20.0000	6.0000
11	1	7.5000	0.0000	3.7500	0.0000	0.0000	20.0000	6.0000
12	1	0.0000	0.0000	15.0000	0.0000	0.0000	20.0000	6.0000
13	1	0.0000	0.0000	15.0000	0.0000	0.0000	20.0000	0.0000
14	1	0.0000	0.0000	15.0000	0.0000	0.0000	20.0000	0.0000
15	1	0.0000	0.0000	15.0000	0.0000	0.0000	20.0000	7.5000
16	1	0.0000	0.0000	12.5000	0.0000	0.0000	20.0000	7.5000
17	1	0.0000	0.0000	20.0000	0.0000	0.0000	11.2500	7.5000
18	1	0.0000	0.0000	20.0000	0.0000	0.0000	15.0000	7.5000
19	1	0.0000	17.5000	0.0000	0.0000	0.0000	15.0000	7.5000
20	1	0.0000	17.5000	0.0000	0.0000	0.0000	15.0000	7.5000
21	1	5.0000	17.5000	0.0000	0.0000	0.0000	7.5000	7.5000
22	1	20.0000	17.5000	0.0000	0.0000	0.0000	0.0000	7.5000
23	1	20.0000	15.2500	0.0000	0.0000	0.0000	0.0000	7.5000
24	1	20.0000	13.0000	0.0000	0.0000	0.0000	0.0000	7.5000
25	1	20.0000	13.0000	0.0000	0.0000	0.0000	0.0000	7.5000
26	1	20.0000	13.0000	0.0000	0.0000	0.0000	0.0000	7.5000
27	1	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000	3.7500
28	1	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
29	1	10.0000	0.0000	0.0000	0.0000	5.6250	0.0000	0.0000
30	1	0.0000	0.0000	0.0000	0.0000	22.5000	0.0000	0.0000
31	1	0.0000	0.0000	0.0000	0.0000	22.5000	0.0000	0.0000
32	1	0.0000	0.0000	0.0000	0.0000	0.0000	16.8750	0.0000
33	1	0.0000	0.0000	0.0000	0.0000	0.0000	22.5000	0.0000
34	1	0.0000	0.0000	0.0000	0.0000	0.0000	22.5000	0.0000
35	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
36	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
37	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
38	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
39	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
40	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Figure 7.10: T21: Last schedule for the packing lines

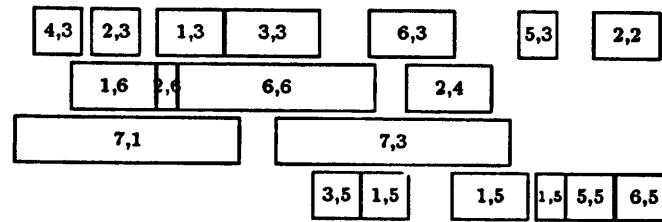


Table E2 T21: Demand corresponding to the last packing lines schedule

1	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	4.5000
2	1	0.0000	0.0000	0.0000	11.2500	0.0000	0.0000	6.0000
3	1	0.0000	0.0000	0.0000	15.0000	0.0000	0.0000	6.0000
4	1	15.0000	0.0000	0.0000	7.5000	0.0000	0.0000	6.0000
5	1	20.0000	11.2500	0.0000	0.0000	0.0000	0.0000	6.0000
6	1	20.0000	15.0000	0.0000	0.0000	0.0000	0.0000	6.0000
7	1	20.0000	7.5000	0.0000	0.0000	0.0000	0.0000	6.0000
8	1	13.7500	5.0000	0.0000	0.0000	0.0000	0.0000	6.0000
9	1	15.0000	10.0000	0.0000	0.0000	0.0000	5.0000	6.0000
10	1	15.0000	0.0000	0.0000	0.0000	0.0000	20.0000	6.0000
11	1	15.0000	0.0000	0.0000	0.0000	0.0000	20.0000	6.0000
12	1	0.0000	0.0000	11.2500	0.0000	0.0000	20.0000	6.0000
13	1	0.0000	0.0000	15.0000	0.0000	0.0000	20.0000	0.0000
14	1	0.0000	0.0000	15.0000	0.0000	0.0000	20.0000	0.0000
15	1	0.0000	0.0000	15.0000	0.0000	0.0000	20.0000	7.5000
16	1	0.0000	0.0000	15.0000	0.0000	0.0000	20.0000	7.5000
17	1	0.0000	0.0000	15.0000	0.0000	0.0000	20.0000	7.5000
18	1	0.0000	0.0000	20.0000	0.0000	0.0000	20.0000	7.5000
19	1	5.0000	0.0000	10.0000	0.0000	0.0000	20.0000	7.5000
20	1	20.0000	0.0000	0.0000	0.0000	0.0000	11.2500	7.5000
21	1	20.0000	0.0000	0.0000	0.0000	0.0000	15.0000	7.5000
22	1	0.0000	17.5000	0.0000	0.0000	0.0000	15.0000	7.5000
23	1	0.0000	17.5000	0.0000	0.0000	0.0000	15.0000	7.5000
24	1	10.0000	17.5000	0.0000	0.0000	0.0000	7.5000	7.5000
25	1	20.0000	17.5000	0.0000	0.0000	0.0000	0.0000	7.5000
26	1	20.0000	8.7500	0.0000	0.0000	0.0000	0.0000	7.5000
27	1	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000	3.7500
28	1	10.0000	0.0000	0.0000	0.0000	11.2500	0.0000	0.0000
29	1	20.0000	0.0000	0.0000	0.0000	15.0000	0.0000	0.0000
30	1	10.0000	0.0000	0.0000	0.0000	5.6250	0.0000	0.0000
31	1	0.0000	0.0000	0.0000	0.0000	22.5000	0.0000	0.0000
32	1	0.0000	13.0000	0.0000	0.0000	22.5000	0.0000	0.0000
33	1	0.0000	13.0000	0.0000	0.0000	0.0000	16.8750	0.0000
34	1	0.0000	13.0000	0.0000	0.0000	0.0000	22.5000	0.0000
35	1	0.0000	6.5000	0.0000	0.0000	0.0000	22.5000	0.0000
36	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
37	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
38	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
39	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
40	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 7.11

T21: Production schedule

Tp	M1		M2		M3		M4	
	Pr	Amt	Pr	Amt	Pr	Amt	Pr	Amt
1	1	15.	4	15.	0	0.	7	7.5
2	1	15.	4	25.	0	0.	7	7.5
3	1	5.	0	0.	0	0.	7	7.5
4	1	15.	2	15.	0	0.	7	7.5
5	1	15.	2	25.	0	0.	7	7.5
6	1	15.	2	25.	6	15.	7	7.5
7	1	15.	2	25.	6	25.	7	7.5
8	1	15.	2	10.	6	5.	7	7.5
9	1	15.	2	10.	6	5.	7	7.5
10	1	15.	0	0.	6	20.	7	7.5
11	1	15.	3	15.	6	20.	7	7.5
12	1	10.	3	25.	6	20.	7	7.5
13	0	0.	3	18.25	6	20.	0	0.
14	0	0.	3	15.	6	20.	0	0.
15	0	0.	3	15.	6	20.	7	7.5
16	0	0.	3	15.	6	20.	7	7.5
17	0	0.	3	15.	6	20.	7	7.5
18	0	0.	3	20.	6	20.	7	7.5
19	1	8.75	3	10.	6	20.	7	7.5
20	1	15.	0	0.	6	10.	7	7.5
21	1	15.	0	0.	6	16.25	7	7.5
22	0	0.	2	15.	6	15.	7	7.5
23	1	10.	2	20.	6	15.	7	7.5
24	1	10.	2	18.25	6	7.5	7	7.5
25	1	15.	2	17.5	0	0.	7	7.5
26	1	15.	2	8.75	0	0.	7	7.5
27	1	15.	5	15.	0	0.	7	3.75
28	1	15.	5	25.	0	0.	0	0.
29	1	15.	5	17.25	0	0.	0	0.
30	1	15.	5	5.625	0	0.	0	0.
31	1	10.	5	20.	0	0.	0	0.
32	0	0.	5	25.	0	0.	0	0.
33	0	0.	0	0.	6	15.	0	0.
34	0	0.	0	0.	6	20.	0	0.
35	0	0.	0	0.	6	25.	0	0.
36	0	0.	0	0.	0	0.	0	0.
37	0	0.	0	0.	0	0.	0	0.
38	0	0.	0	0.	0	0.	0	0.
39	0	0.	0	0.	0	0.	0	0.
40	0	0.	0	0.	0	0.	0	0.

Figure 7.11 T22: First schedule for the packing lines

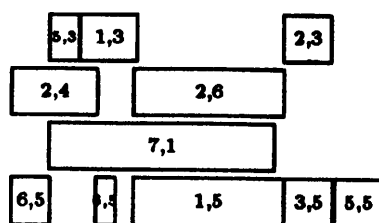


Table F1 T22: Demand corresponding to the first packing lines schedule

1	1	0	0.0000	16	1250	0	0.0000	0.0000	0.0000	16.8750	0.0000
2	1	0	0.0000	17	5000	0	0.0000	0.0000	0.0000	22.5000	0.0000
3	1	0	0.0000	17	5000	0	0.0000	0.0000	0.0000	15.0000	0.0000
4	1	3	0.0000	17	5000	0	0.0000	0.0000	7.5000	0.0000	6.0000
5	1	15	0.0000	8	7500	0	0.0000	0.0000	0.0000	0.0000	6.0000
6	1	15	0.0000	0	0.0000	0	0.0000	0.0000	0.0000	11.2500	6.0000
7	1	12	0.0000	11	0.0000	0	0.0000	0.0000	0.0000	11.2500	6.0000
8	1	20	0.0000	20	0.0000	0	0.0000	0.0000	0.0000	0.0000	6.0000
9	1	30	0.0000	20	0.0000	0	0.0000	0.0000	0.0000	0.0000	6.0000
10	1	20	0.0000	20	0.0000	0	0.0000	0.0000	0.0000	0.0000	6.0000
11	1	20	0.0000	20	0.0000	0	0.0000	0.0000	0.0000	0.0000	6.0000
12	1	20	0.0000	20	0.0000	0	0.0000	0.0000	0.0000	0.0000	6.0000
13	1	20	0.0000	20	0.0000	0	0.0000	0.0000	0.0000	0.0000	6.0000
14	1	20	0.0000	20	0.0000	0	0.0000	0.0000	0.0000	0.0000	6.0000
15	1	10	0.0000	12	7500	5	0.0000	0.0000	0.0000	0.0000	0.0000
16	1	0	0.0000	15	0.0000	20	0.0000	0.0000	0.0000	0.0000	0.0000
17	1	0	0.0000	15	0.0000	20	0.0000	0.0000	0.0000	0.0000	0.0000
18	1	0	0.0000	0	0.0000	0	0.0000	0.0000	16.8750	0.0000	0.0000
19	1	0	0.0000	0	0.0000	0	0.0000	0.0000	22.5000	0.0000	0.0000
20	1	0	0.0000	0	0.0000	0	0.0000	0.0000	11.2500	0.0000	0.0000
21	1	0	0.0000	0	0.0000	0	0.0000	0.0000	0.0000	0.0000	0.0000
22	1	0	0.0000	0	0.0000	0	0.0000	0.0000	0.0000	0.0000	0.0000
23	1	0	0.0000	0	0.0000	0	0.0000	0.0000	0.0000	0.0000	0.0000
24	1	0	0.0000	0	0.0000	0	0.0000	0.0000	0.0000	0.0000	0.0000
25	1	0	0.0000	0	0.0000	0	0.0000	0.0000	0.0000	0.0000	0.0000
26	1	0	0.0000	0	0.0000	0	0.0000	0.0000	0.0000	0.0000	0.0000
27	1	0	0.0000	0	0.0000	0	0.0000	0.0000	0.0000	0.0000	0.0000
28	1	0	0.0000	0	0.0000	0	0.0000	0.0000	0.0000	0.0000	0.0000
29	1	0	0.0000	0	0.0000	0	0.0000	0.0000	0.0000	0.0000	0.0000
30	1	0	0.0000	0	0.0000	0	0.0000	0.0000	0.0000	0.0000	0.0000
31	1	0	0.0000	0	0.0000	0	0.0000	0.0000	0.0000	0.0000	0.0000
32	1	0	0.0000	0	0.0000	0	0.0000	0.0000	0.0000	0.0000	0.0000
33	1	0	0.0000	0	0.0000	0	0.0000	0.0000	0.0000	0.0000	0.0000
34	1	0	0.0000	0	0.0000	0	0.0000	0.0000	0.0000	0.0000	0.0000
35	1	0	0.0000	0	0.0000	0	0.0000	0.0000	0.0000	0.0000	0.0000
36	1	0	0.0000	0	0.0000	0	0.0000	0.0000	0.0000	0.0000	0.0000
37	1	0	0.0000	0	0.0000	0	0.0000	0.0000	0.0000	0.0000	0.0000
38	1	0	0.0000	0	0.0000	0	0.0000	0.0000	0.0000	0.0000	0.0000
39	1	0	0.0000	0	0.0000	0	0.0000	0.0000	0.0000	0.0000	0.0000
40	1	0	0.0000	0	0.0000	0	0.0000	0.0000	0.0000	0.0000	0.0000

Figure 7.12 : T22: Last schedule for the packing lines

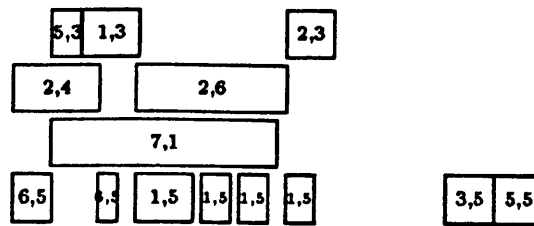


Table F2 T22: Demand corresponding to the last packing lines schedule

1	1	0.0000	13.1250	0.0000	0.0000	0.0000	16.8750	0.0000
2	1	0.0000	17.5000	0.0000	0.0000	0.0000	22.5000	0.0000
3	1	0.0000	17.5000	0.0000	0.0000	15.0000	0.0000	6.0000
4	1	3.7500	17.5000	0.0000	0.0000	7.5000	0.0000	6.0000
5	1	15.0000	8.7500	0.0000	0.0000	0.0000	11.2500	6.0000
6	1	15.0000	0.0000	0.0000	0.0000	0.0000	11.2500	6.0000
7	1	12.5000	10.0000	0.0000	0.0000	0.0000	0.0000	6.0000
8	1	20.0000	20.0000	0.0000	0.0000	0.0000	0.0000	6.0000
9	1	20.0000	20.0000	0.0000	0.0000	0.0000	0.0000	6.0000
10	1	10.0000	20.0000	0.0000	0.0000	0.0000	0.0000	6.0000
11	1	20.0000	20.0000	0.0000	0.0000	0.0000	0.0000	6.0000
12	1	10.0000	20.0000	0.0000	0.0000	0.0000	0.0000	6.0000
13	1	20.0000	20.0000	0.0000	0.0000	0.0000	0.0000	6.0000
14	1	10.0000	20.0000	0.0000	0.0000	0.0000	0.0000	6.0000
15	1	20.0000	13.7500	0.0000	0.0000	0.0000	0.0000	0.0000
16	1	20.0000	15.0000	0.0000	0.0000	0.0000	0.0000	0.0000
17	1	0.0000	15.0000	0.0000	0.0000	0.0000	0.0000	0.0000
18	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
19	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
20	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
21	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
22	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
23	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
24	1	0.0000	0.0000	15.0000	0.0000	0.0000	0.0000	0.0000
25	1	0.0000	0.0000	20.0000	0.0000	0.0000	0.0000	0.0000
26	1	0.0000	0.0000	10.0000	0.0000	5.6250	0.0000	0.0000
27	1	0.0000	0.0000	0.0000	0.0000	22.5000	0.0000	0.0000
28	1	0.0000	0.0000	0.0000	0.0000	22.5000	0.0000	0.0000
29	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
30	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
31	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
32	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
33	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
34	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
35	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
36	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
37	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
38	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
39	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
40	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 7.12

T22: Production schedule

Tp	M1		M2		M3		M4	
	Pr	Amt	Pr	Amt	Pr	Amt	Pr	Amt
1	0	0.	2	15.	6	18.75	0	0.
2	0	0.	2	25.	6	20.	0	0.
3	0	0.	2	25.	0	0.	7	5.
4	0	0.	2	25.	0	0.	7	7.
5	1	15.	0	0.	0	0.	7	5.
6	1	15.	2	15.	0	0.	7	7.
7	1	15.	2	20.	0	0.	7	5.
8	1	15.	2	20.	0	0.	7	7.
9	1	15.	2	20.	0	0.	7	5.
10	1	15.	2	20.	0	0.	7	7.
11	1	15.	2	20.	0	0.	7	5.
12	1	15.	2	20.	0	0.	7	7.
13	1	15.	2	20.	0	0.	7	5.
14	1	15.	2	20.	0	0.	7	7.
15	1	15.	2	18.625	0	0.	0	0.
16	1	15.	2	15.	0	0.	0	0.
17	1	15.	2	15.	0	0.	0	0.
18	0	0.	0	0.	0	0.	0	0.
19	0	0.	0	0.	0	0.	0	0.
20	0	0.	0	0.	0	0.	0	0.
21	0	0.	0	0.	0	0.	0	0.
22	0	0.	0	0.	0	0.	0	0.
23	0	0.	0	00.	0	0.	0	0.
24	0	0.	3	15.	0	0	0	0.
25	0	0.	5	15.	0	0.	0	0.
26	0	0.	5	10	0	0.	0	0.
27	0	0.	5	25.	0	0.	0	0.
28	0	0.	5	20.	0	0.	0	0.
29	0	0.	0	0	0	0.	0	0.
30	0	0.	0	0	0	0.	0	0.
31	0	0.	0	0.	0	0.	0	0.
32	0	0.	0	0.	0	0.	0	0.
33	0	0.	0	0.	0	0.	0	0.
34	0	0.	0	0.	0	0.	0	0.
35	0	0.	0	0.	0	0.	0	0.
36	0	0.	0	0.	0	0.	0	0.
37	0	0.	0	0.	0	0.	0	0.
38	0	0.	0	0.	0	0.	0	0.
39	0	0.	0	0.	0	0.	0	0.
40	0	0.	0	0.	0	0.	0	0.

Figure 7.13: T23: First schedule for the packing lines

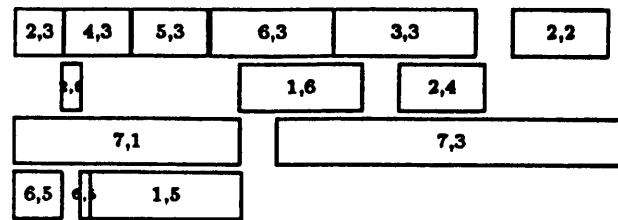


Table G1 T23: Demand corresponding to the first packing lines schedule

1	1	0.0000	11.2500	0.0000	0.0000	0.0000	16.8750	4.5000
2	1	0.0000	15.0000	0.0000	0.0000	0.0000	22.5000	6.0000
3	1	0.0000	12.5000	0.0000	3.7500	0.0000	11.2500	6.0000
4	1	0.0000	10.0000	0.0000	15.0000	0.0000	11.2500	6.0000
5	1	15.0000	0.0000	0.0000	15.0000	0.0000	0.0000	6.0000
6	1	20.0000	0.0000	0.0000	15.0000	0.0000	0.0000	6.0000
7	1	20.0000	0.0000	0.0000	0.0000	11.2500	0.0000	6.0000
8	1	20.0000	0.0000	0.0000	0.0000	15.0000	0.0000	6.0000
9	1	20.0000	0.0000	0.0000	0.0000	15.0000	0.0000	6.0000
10	1	20.0000	0.0000	0.0000	0.0000	15.0000	0.0000	6.0000
11	1	20.0000	0.0000	0.0000	0.0000	0.0000	11.2500	6.0000
12	1	20.0000	0.0000	0.0000	0.0000	0.0000	15.0000	6.0000
13	1	15.0000	0.0000	0.0000	0.0000	0.0000	15.0000	0.0000
14	1	20.0000	0.0000	0.0000	0.0000	0.0000	15.0000	0.0000
15	1	20.0000	0.0000	0.0000	0.0000	0.0000	15.0000	7.5000
16	1	20.0000	0.0000	0.0000	0.0000	0.0000	15.0000	7.5000
17	1	20.0000	0.0000	3.7500	0.0000	0.0000	7.5000	7.5000
18	1	20.0000	0.0000	15.0000	0.0000	0.0000	0.0000	7.5000
19	1	10.0000	0.0000	15.0000	0.0000	0.0000	0.0000	7.5000
20	1	0.0000	0.0000	15.0000	0.0000	0.0000	0.0000	7.5000
21	1	0.0000	8.7500	15.0000	0.0000	0.0000	0.0000	7.5000
22	1	0.0000	17.5000	15.0000	0.0000	0.0000	0.0000	7.5000
23	1	0.0000	17.5000	15.0000	0.0000	0.0000	0.0000	7.5000
24	1	0.0000	17.5000	15.0000	0.0000	0.0000	0.0000	7.5000
25	1	0.0000	17.5000	0.0000	0.0000	0.0000	0.0000	7.5000
26	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	7.5000
27	1	0.0000	13.0000	0.0000	0.0000	0.0000	0.0000	7.5000
28	1	0.0000	13.0000	0.0000	0.0000	0.0000	0.0000	7.5000
29	1	0.0000	13.0000	0.0000	0.0000	0.0000	0.0000	7.5000
30	1	0.0000	13.0000	0.0000	0.0000	0.0000	0.0000	7.5000
31	1	0.0000	13.0000	0.0000	0.0000	0.0000	0.0000	7.5000
32	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	7.5000
33	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	3.7500
34	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
35	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
36	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
37	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
38	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
39	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
40	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Figure 7.14: T23: Last schedule for the packing lines

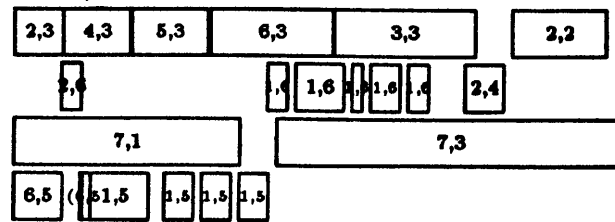


Table G2 T23: Demand corresponding to the last packing lines schedule

1	1	0.0000	11.2500	0.0000	0.0000	0.0000	16.8750	4.5000
2	1	0.0000	15.0000	0.0000	0.0000	0.0000	22.5000	6.0000
3	1	0.0000	12.5000	0.0000	3.7500	0.0000	11.2500	6.0000
4	1	0.0000	10.0000	0.0000	15.0000	0.0000	11.2500	6.0000
5	1	15.0000	0.0000	0.0000	15.0000	0.0000	0.0000	6.0000
6	1	20.0000	0.0000	0.0000	15.0000	0.0000	0.0000	6.0000
7	1	20.0000	0.0000	0.0000	0.0000	11.2500	0.0000	6.0000
8	1	10.0000	0.0000	0.0000	0.0000	15.0000	0.0000	6.0000
9	1	20.0000	0.0000	0.0000	0.0000	15.0000	0.0000	6.0000
10	1	10.0000	0.0000	0.0000	0.0000	15.0000	0.0000	6.0000
11	1	20.0000	0.0000	0.0000	0.0000	0.0000	11.2500	6.0000
12	1	10.0000	0.0000	0.0000	0.0000	0.0000	15.0000	6.0000
13	1	20.0000	0.0000	0.0000	0.0000	0.0000	15.0000	0.0000
14	1	15.0000	0.0000	0.0000	0.0000	0.0000	15.0000	0.0000
15	1	10.0000	0.0000	0.0000	0.0000	0.0000	15.0000	7.5000
16	1	20.0000	0.0000	0.0000	0.0000	0.0000	15.0000	7.5000
17	1	20.0000	0.0000	3.7500	0.0000	0.0000	7.5000	7.5000
18	1	10.0000	0.0000	15.0000	0.0000	0.0000	0.0000	7.5000
19	1	10.0000	0.0000	15.0000	0.0000	0.0000	0.0000	7.5000
20	1	20.0000	0.0000	15.0000	0.0000	0.0000	0.0000	7.5000
21	1	10.0000	0.0000	15.0000	0.0000	0.0000	0.0000	7.5000
22	1	20.0000	0.0000	15.0000	0.0000	0.0000	0.0000	7.5000
23	1	0.0000	0.0000	15.0000	0.0000	0.0000	0.0000	7.5000
24	1	0.0000	0.0000	15.0000	0.0000	0.0000	0.0000	7.5000
25	1	0.0000	17.5000	0.0000	0.0000	0.0000	0.0000	7.5000
26	1	0.0000	17.5000	0.0000	0.0000	0.0000	0.0000	7.5000
27	1	0.0000	13.0000	0.0000	0.0000	0.0000	0.0000	7.5000
28	1	0.0000	13.0000	0.0000	0.0000	0.0000	0.0000	7.5000
29	1	0.0000	13.0000	0.0000	0.0000	0.0000	0.0000	7.5000
30	1	0.0000	13.0000	0.0000	0.0000	0.0000	0.0000	7.5000
31	1	0.0000	13.0000	0.0000	0.0000	0.0000	0.0000	7.5000
32	1	0.0000	17.5000	0.0000	0.0000	0.0000	0.0000	7.5000
33	1	0.0000	17.5000	0.0000	0.0000	0.0000	0.0000	3.7500
34	1	0.0000	9.7500	0.0000	0.0000	0.0000	0.0000	0.0000
35	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
36	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
37	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
38	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
39	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
40	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 7.13. T23: Production schedule

Tp	M1		M1		M1		M1	
	Pr	Amt	Pr	Amt	Pr	Amt	Pr	Amt
1	0	0.	0	0.	6	16.875	7	4.5
2	0	0.	0	0.	6	20.	7	5.
3	0	0.	0	0.	6	10.	7	7.
4	0	0.	4	15.	6	15.	7	5.
5	1	15.	4	18.75	0	0.	7	7.
6	1	15.	4	15.	0	0.	7	5.
7	1	15.	5	10.	0	0.	7	7.
8	1	15.	5	19.	0	0.	7	5.
9	1	15.	5	15.	0	0.	7	7.
10	1	15.	5	15.	0	0.	7	5.
11	1	15.	0	0.	6	10.	7	7.
12	1	15.	0	0.	6	16.25	7	5.
13	1	15.	0	0.	6	15.	0	0.
14	1	15.	0	0.	6	15.	0	0.
15	1	15.	0	0.	6	15.	7	7.5
16	1	15.	0	0.	6	15.	7	7.5
17	1	15.	0	0.	6	7.5	7	7.5
18	1	15.	3	15.	0	0.	7	7.5
19	1	15.	3	18.75	0	0.	7	7.5
20	1	15.	3	15.	0	0.	7	7.5
21	1	15.	3	15.	0	0.	7	7.5
22	1	15.	3	15.	0	0.	7	7.5
23	1	15.	0	0.	0	0.	7	7.5
24	0	0.	2	15.	0	0.	7	7.5
25	0	0.	2	25.	0	0.	7	7.5
26	0	0.	2	25.	0	0.	7	7.5
27	0	0.	2	25.	0	0.	7	7.5
28	0	0.	2	19.75	0	0.	7	7.5
29	0	0.	2	10.	0	0.	7	7.5
30	0	0.	2	16.	0	0.	7	7.5
31	0	0.	2	10.	0	0.	7	7.5
32	0	0.	2	20.	0	0.	7	7.5
33	0	0.	2	18.	0	0.	7	4.75
34	0	0.	2	8.75	0	0.	0	0.
35	0	0.	0	0.	0	0.	0	0.
36	0	0.	0	0.	0	0.	0	0.
37	0	0.	0	0.	0	0.	0	0.
38	0	0.	0	0.	0	0.	0	0.
39	0	0.	0	0.	0	0.	0	0.
40	0	0.	0	0.	0	0.	0	0.

Figure 7.15: T24: First schedule for the packing lines

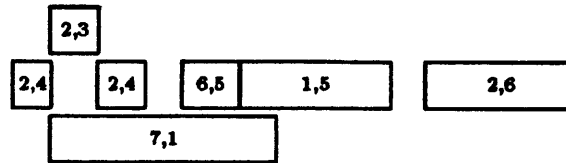


Table H1 T24: Demand corresponding to the first packing lines schedule

1	1	0.0000	13	1250	0.0000	0.0000	0.0000	0.0000	0.0000
2	1	0.0000	17	5000	0.0000	0.0000	0.0000	0.0000	0.0000
3	1	0.0000	15	0000	0.0000	0.0000	0.0000	0.0000	6.0000
4	1	0.0000	15	0000	0.0000	0.0000	0.0000	0.0000	6.0000
5	1	0.0000	16	2500	0.0000	0.0000	0.0000	0.0000	6.0000
6	1	0.0000	17	5000	0.0000	0.0000	0.0000	0.0000	6.0000
7	1	0.0000	17	5000	0.0000	0.0000	0.0000	0.0000	6.0000
8	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	6.0000
9	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	6.0000
10	1	0.0000	0.0000	0.0000	0.0000	0.0000	22.5000	6.0000	6.0000
11	1	0.0000	0.0000	0.0000	0.0000	0.0000	22.5000	6.0000	6.0000
12	1	0.0000	0.0000	0.0000	0.0000	0.0000	22.5000	6.0000	6.0000
13	1	15.0000	0.0000	0.0000	0.0000	0.0000	0.0000	6.0000	6.0000
14	1	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000	6.0000	6.0000
15	1	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
16	1	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
17	1	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
18	1	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
19	1	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
20	1	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
21	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
22	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
23	1	0.0000	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
24	1	0.0000	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
25	1	0.0000	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
26	1	0.0000	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
27	1	0.0000	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
28	1	0.0000	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
29	1	0.0000	21.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
30	1	0.0000	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
31	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
32	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
33	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
34	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
35	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
36	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
37	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
38	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
39	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
40	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Figure 7.16: T24: Last schedule for the packing lines

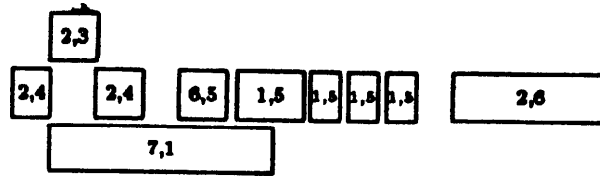


Table H2 T24: Demand corresponding to the last packing lines schedule

1	1	0.0000	13.1250	0.0000	0.0000	0.0000	0.0000	0.0000
2	1	0.0000	17.5000	0.0000	0.0000	0.0000	0.0000	0.0000
3	1	0.0000	15.0000	0.0000	0.0000	0.0000	0.0000	6.0000
4	1	0.0000	15.0000	0.0000	0.0000	0.0000	0.0000	6.0000
5	1	0.0000	16.2500	0.0000	0.0000	0.0000	0.0000	6.0000
6	1	0.0000	17.5000	0.0000	0.0000	0.0000	0.0000	6.0000
7	1	0.0000	17.5000	0.0000	0.0000	0.0000	0.0000	6.0000
8	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	6.0000
9	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	6.0000
10	1	0.0000	0.0000	0.0000	0.0000	0.0000	22.5000	6.0000
11	1	0.0000	0.0000	0.0000	0.0000	0.0000	22.5000	6.0000
12	1	0.0000	0.0000	0.0000	0.0000	0.0000	22.5000	6.0000
13	1	15.0000	0.0000	0.0000	0.0000	0.0000	0.0000	6.0000
14	1	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000	6.0000
15	1	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
16	1	10.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
17	1	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
18	1	10.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
19	1	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
20	1	10.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
21	1	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
22	1	10.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
23	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
24	1	0.0000	10.0000	0.0000	0.0000	0.0000	0.0000	0.0000
25	1	0.0000	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000
26	1	0.0000	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000
27	1	0.0000	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000
28	1	0.0000	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000
29	1	0.0000	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000
30	1	0.0000	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000
31	1	0.0000	20.0000	0.0000	0.0000	0.0000	0.0000	0.0000
32	1	0.0000	10.0000	0.0000	0.0000	0.0000	0.0000	0.0000
33	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
34	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
35	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
36	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
37	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
38	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
39	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
40	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 7.14. Production schedule

Tp	M1		M1		M1		M1	
	Pr	Amt	Pr	Amt	Pr	Amt	Pr	Amt
1	0	0.	2	10.	0	0.	0	0.
2	0	0.	2	2.	0	0.	0	0.
3	0	0.	2	15.625	0	0.	7	5.
4	0	0.	2	15.	0	0.	7	7.
5	0	0.	2	16.25	0	0.	7	5.
6	0	0.	2	17.5	0	0.	7	7.
7	0	0.	2	17.5	0	0.	7	5.
8	0	0.	0	0.	0	0.	7	7.
9	0	0.	0	0.	0	0.	7	5.
10	0	0.	0	0.	6	15.	7	7.
11	0	0.	0	0.	6	25.	7	5.
12	0	0.	0	0.	6	25.	7	7.
13	1	15.	0	0.	0	0.	7	5.
14	1	15.	0	0.	0	0.	7	7.
15	1	15.	0	0.	0	0.	0	0.
16	1	15.	0	0.	0	0.	0	0.
17	1	15.	0	0.	0	0.	0	0.
18	1	15.	0	0.	0	0.	0	0.
19	1	15.	0.	0	0.	0	0	0
20	1	15.	0	0.	0	0.	0	0.
21	1	15.	0	0.	0	0.	0	0.
22	1	15.	0	0.	0	0.	0	0.
23	1	6.25	0	00.	0	0.	0	0.
24	0	0.	3	15.	2	10.	0	0.
25	0	0.	5	15.	2	20.	0	0.
26	0	0.	5	10.	2	20.	0	0.
27	0	0.	5	25.	2	20.	0	0.
28	0	0.	5	20.	2	20.	0	0.
29	0	0.	0	0	2	20.	0	0.
30	0	0.	0	0	2	20.	0	0.
31	0	0.	0	0.	2	20.	0	0.
32	0	0.	0	0.	2	10.	0	0.
33	0	0.	0	0.	0	0.	0	0.
34	0	0.	0	0.	0	0.	0	0.
35	0	0.	0	0.	0	0.	0	0.
36	0	0.	0	0.	0	0.	0	0.
37	0	0.	0	0.	0	0.	0	0.
38	0	0.	0	0.	0	0.	0	0.
39	0	0.	0	0.	0	0.	0	0.
40	0	0.	0	0.	0	0.	0	0.

Chapter 8

Conclusion

In this thesis, the short term production scheduling for a two stage manufacturing system was considered. The system consists of a number of manufacturing facilities followed by a number of packaging facilities. Before packing the products may be stored temporarily (in process inventory) in a number of silos. The problem was to minimise packing and changeover costs at the packing lines and the number of set ups at the manufacturing units. The minimum run length and the efficient use of the intermediate storage facilities were significant constraints at the manufacturing level.

A review of the literature did not reveal a wealth of publications on the subject, therefore a novel approach had to be investigated.

A stage by stage multipass method was developed. The multipass procedure is interactive, in that it is conducted by the user. The packing lines schedule is first built, then the manufacturing units schedule is determined with the packing lines schedule as demand.

The packing lines system consists of a number of interdependent machines. The interdependence is introduced by shared resources among the lines. The problem was to minimise changeover and packing costs. Due dates for all items fall at the end of the horizon. In a first attempt, the problem was formulated with preemption as a linear integer problem. However, due to the size of the resulting model, this formulation was considered infeasible. New avenues had

to be explored. A new formulation whereby the demand for each item was split in smaller sublots was considered. The problem was to schedule these sublots without preemption.

A branch and bound algorithm was constructed. In a first step, the resource constraints were dropped and a relaxation of the resulting problem was proposed for computing lower bounds. This relaxation decouples the problem in two problems: a machine loading, formulated as a general assignment problem and a sequencing problem, formulate as a shortest arborescence problem. In order to strengthen the bounds, penalties were computed.

Since the resource constraints elongate a schedule, at every node of the tree, among the items belonging to the same class, the one introducing the minimum idle time while satisfying the resource constraints is selected. This selection is carried out within an explicit enumeration procedure. The method worked well for problems with an even distribution of products among the sizes, that is where the number and quality of items are approximatively the same for all classes. The computational effort depends on the depth of the tree, near the root, particularly for large problems, it is quite heavy, the deeper the search, the lighter it is. As the resource constraints become tighter, the gap between the lower bound and the optimal solution widens. This is confirmed by the computational results. Nevertheless, in general, feasible solutions were obtained quickly. However, although these solutions seemed to be optimal in a number of cases, optimality was not proved after a substantial number of nodes. This is a general feature of tree search with poor bounding. In this respect, it is worthwhile investigating more the use of the penalties.

On the other hand, better feasibility tests should be constructed to ensure that the search does not drift away from the direction of feasible solutions. Sometimes, near the root, the assignment problem takes a long time to solve to optimality or to prove infeasibility. To alleviate this disadvantage, it would be better to limit the machine loading subproblem to finding a first feasible solution after a number of nodes. The coresponding node in the overall algorithm is

fathomed if no solution is found after the limit.

When the packing lines are dedicated to packing the format of one product, the search becomes no longer exhaustive. For this a priority rule was introduced that makes the procedure more of a heuristic.

As a possible extension of the model, the introduction of due dates can be considered. These can be introduced either in the cost function, by including a high cost of packing a particular item after its due date or in the constraints, by forbidding its packing after its due dates.

A branch and bound algorithm with lagrangean relaxation was developed for the manufacturing units-intermediate storage subproblem. At first, only the case with dedicated storage was considered. The constraints imposing that a machine can produce one product in one time period were dualised. In the resulting one product subproblems, the quantity to produce in one time period was computed using a simple simulator. This was possible since a manufacturing unit can handle at most one product at a time. A dynamic programming procedure was used to find the optimal allocation of the manufacturing units to the products. The procedure was made faster by taking advantage of the simplicity of the cost function.

The formulation was extended to the flexible storage case. In this case, the constraints ensuring that the number of silos allocated to all products, at a given time period, should not exceed the number of silos available, was dualised as well. The same solution procedure was used.

In the experiments carried out, the minimum run length constraint was expressed in terms of the quantity produced during the period where the manufacturing unit is set up to the current product. For the flexible storage case, instead of pricing silos allocation, it was introduced in terms of priority in the dynamic programming procedure. A number of priorities were tested and the best results were obtained when priority was given to allocating silos rather than not to.

It was rare that overall feasibility was obtained at the first pass, except for

very simple problems not reported in this thesis. Coordination was always conducted by the user. The higher the demand, the more difficult overall feasibility was to achieve. For some problems however, it seems that infeasibility was due to overcapacity demand imposed by the long term scheduler.

As long as the packing lines subproblem was not overconstrained, the method gave good results since it was possible to carry out more passes before infeasibility was encountered. Optimality of the packing lines schedule obtained at the first iteration was conserved and it was rather a matter of delaying the packing of some products to allow for production and resolve bottlenecks. When the packing lines subproblem is tightly constrained and the demand is high, infeasibility was obtained early and it was necessary to alter the lot sizes.

Except in two or three instances, the minimum run length constraints for the manufacturing units was always satisfied. In these cases, this was due to small isolated demands that could not be satisfied from storage. To alleviate the infeasibility, these demands were simply deleted since they were negligible.

An area worth exploring is the automatisisation of the coordination device. The work could be based on the expertise acquired by the user of the interactive device. It should be done separately for every plant. Indeed, as it may have been noticed, the coordination procedure depends very much on the structure and operational data of the plant. It was easier for the second plant where there were four manufacturing units, dedicated silos and a few packing lines. We believe, though, that a degree of interaction should be introduced at some stage, allowing for flexibility and robustness. Indeed, for some very constrained problems, it may very quickly degenerate and will diverge from any feasibility path.

References

- [1] BAKER K.R. and SCHRAGE L.E., (1978), Finding an optimal sequence by dynamic programming: An extension to precedence related tasks. *Opns. Res.* 26, pp. 111-120.
- [2] BAKER K. R., DIXON P., MAGAZINE M. J. and SILVER E. A., (1978), An algorithm for the dynamic lot size problem with time varying production constraints. *Mgmt. Sc.* 24, pp.1710-1720.
- [3] R.BELLMAN, (1957), *Dynamic programming* Priceton University Press.
- [4] CHALMET L., GELDERS L., (1977), Lagrangean relaxation for a generalised assignment, type problem. In *Advances in Operations Research* Marc Roubens editor, North Holland, pp. 103-109.
- [5] CHU YOENG-JIN and LIU TSENG-HONG, (1965), On the shortest arborescence of a directed graph. *Scientia sinica* XIV, pp. 1396-1400.
- [6] CROWSTON W. B. and WAGNER M. H., (1973), Dynamic lot size models for multistage assembly systems. *Mgmt. Sc.* 20, pp. 14-21.
- [7] DANTZIG G. B., FULKERSON D. R. and JOHNSON S. M., (1959), On a linear programming, combinatorial approach to the travelling salesman. *Opns. Res.* 7, pp. 58-66.
- [8] DANTZIG G. B., (1957), Discrete variables extremum problems. *Opns. Res.* 5, pp. 266-277.
- [9] DAVIES E.W., (1973), Project scheduling under resource constraints: A

- historical review and categorisation of the procedures. *A.I.I.E. Transactions* 5, pp. 297-313.
- [10] DORSEY R. C., HODGSON T. J. and RATLIFF H. D., (1975), A network approach to a multifacility multiproduct production scheduling problem without backordering. *Mgmt. Sc.* 21, pp. 813-822.
 - [11] DZIELINSKI B. and GOMORY R., (1965), Optimal programming of lot sizes, inventories and labour allocations. *Mgmt Sc.* 11, pp. 874-890.
 - [12] EASTMAN W.L., EVEN S. ISAACS I.M., (1964), Bounds for the optimal scheduling of n jobs on m processors. *Mgmt. Sc.* 11, pp. 268-279.
 - [13] EDMONDS J., (1967), Optimum branchings. *J. Res. Nat. Bur. Standards* 71B, pp.233-240.
 - [14] ELMAGHRABY S. E., (1968), The machine sequencing problem: Review and extensions. *Nav. Res. Log. Quart.* 15, No. 2, pp. 205-232
 - [15] ELMAGHRABY S. E., MALLIK A. K. and NUTTLE H. L. W., (1970), The scheduling of lots on a single facility. *AIIE Transactions*, 2, pp. 203-213.
 - [16] FISHER M.L., (1973), Optimal solution of scheduling problems using lagrange multipliers Part I.. *Opns. Res.* 21, pp. 1114-1127.
 - [17] FISHER M.L., (1976), A dual algorithm for the one machine scheduling problem. *Math. Programming* 11, pp. 701-715
 - [18] FISHER M. L., (1981), The Lagrangean relaxation method for solving integer programming problems. *Mgmt. Sc.* 27, pp. 1-18.
 - [19] FISHER M. L., JAIKUMAR R., and VAN WASSENHOFE Luk N., (1986), A multipliers adjustment for the generalised assignment problem. *Mgmt. Sc.* 32, pp. 1005-1103.
 - [20] FLORIAN M. and KLEIN M., (1971), Deterministic production planning with concave costs and capacity constraints. *Mgmt. Sc.* 18, pp. 2-20.

- [21] FULKERSON D. R., (1974), Packing rooted directed cuts in a weighted directed graph. *Math. Programming* 6, pp. 1-13.
- [22] GARFINKEL R. S., (1985), The travelling salesman: motivation and modeling. In *The travelling salesman problem* , E. L. Lawler & all editors, John Wiley, pp. 17-36.
- [23] GELDERS L. F., MAES J. and VAN WASSENHOFE Luk N., (1985), A branch and bound algorithm for the multi item single level capacitated dynamic lot sizing. In *Multi stage production planning and inventory control* S. Axsater and al. editors, Springer Verlag, pp. 92-108.
- [24] L. F. GELDERS, L. N. VAN WASSENHOFE, (1981), Production planning: a review. *Eur. J. of Opnl. Res.* 7, pp. 101-110.
- [25] GEOFFRION A. M., (1972), Lagrangean relaxation. *Math. Programming Study* No. 2, pp. 82-114.
- [26] GEOFFRION A. M. and MARSTEN R., (1972), Integer programming algorithms: a framework and state of the art survey. *Mgmt. Sc.* 18, pp. 465-491.
- [27] GEOFFRION A. M. and GRAVES G. W., (1976), Scheduling parallel production lines with changeover considerations. *Opns. Res.* 24, pp. 595-610.
- [28] GILMORE R. C. and GOMORY R. E., (1964), Sequencing a one state variable machine: a solvable case of the travelling salesman. *Opns. Res.* 12, pp. 655-679.
- [29] GOMORY R. E., (1958), Outline of an algorithm for integer solutions to linear programs. *Bull. of the Amer. Math. Soci.* 64, pp. 275-278.
- [30] GLASSEY C. R., (1968), Minimum changeover scheduling on one machine. *Opns. Res.* 16, pp. 343-353.
- [31] GRAVES S. C., (1981), A review of production scheduling. *Opns. Res.*, 29, pp. 646-675.

- [32] GRAVES S. C., (1981), Multi-stage lot sizing: an iterative procedure. *TIMS Studies in Management Sciences* 16, North Holland Publishing Company, pp. 95-110.
- [33] GUNTHER H. O., (1985), The design of a hierarchical model for production planning and scheduling. In *Multi-stage production planning and inventory control*, S. Axsater and al. editors, Springer Verlag, pp. 227-260.
- [34] HANSEN P., (1975), Les procedures d'exploration et d'optimisation par separation et evaluation. In *Combinatorial optimisation: methods and applications* B. Roy editor, D. Reidel Publishing Company, pp. 29-65.
- [35] HAX A. C. and DAN Candea, (1984), *Production and inventory management*. Prentice-Hall.
- [36] HELD M. and KARP R. M., (1970), Traveling salesman and minimum spanning trees. Part 1. *Opns. Res.* 18, pp. 1139-1162.
- [37] HELD M., WOLFE P. and CROWDER H. P., (1974), Validation of sub-gradient optimisation. *Math. Programming* 6, pp. 62-88.
- [38] HORN W.A., (1973), Minimizing average flow time with parallel machines. *Opns. Res.* 21, pp. 846-847.
- [39] IGNALL I. and SCHRAGEL., (1965), Application Of the branch and bound technique to some flow shop scheduling problems. *Math. of Opns. Res.* 3, pp. 197-204.
- [40] INGARGIOLA G. P. and KORSH J. F., (1973), Reduction algorithm for the zero one knapsack problems . *Mgmt. Sc.* 20, (part I), pp. 460-463.
- [41] JAGANNATHAN R. and RAO M. R., (1973), A class of deterministic production planning problems. *Mgmt. Sc.* 19, pp. 1295-1300.
- [42] JOHNSON S.M., (1954), Optimal two and three stages production schedules with set up times included. *Naval Res. Logist. Quart.* 1, pp. 61-68
- [43] KARMAKAR U. and SCHRAGE L., (1985), Dynamic product cycling problem . *Opns. Res.* 33, pp. 326-345.

- [44] KALYMON B. A., (1972), A decomposition algorithm for arborescence inventory systems. *Opns. Res.* 20, pp. 860-873.
- [45] MURTY KATTA G., *linear programming* (1983) John Wiley and sons.
- [46] KIWIEL K. C., An algorithm for linearly constrained convex nondifferentiable minimization procedure. *J. of Math. Analysis and Applic.* , No. 105, pp. 452-465.
- [47] KOOPMANS T. C. and BECKMAN M. J., (1957), Assignment problems and the location of economic activities. *Econometrica* 25, pp. 53-76.
- [48] LAGEWEL B.J., LENSTRA J.K, RINNOY KAN A.H.G., (1977), Job shop scheduling by implicit enumeration. *Mgmt. Sc.* 24, pp. 441-450.
- [49] LAGEWEL B.J., LENSTRA J.K, RINNOY KAN A.H.G., (1978), A general bounding scheme for the permutation flow shop problem. *Opns. Res.* 26, pp.53-67.
- [50] LAMBRECHT M. and VANDER EECKEN J., (1978), A capacity constrained single facility dynamic lot size model. *Eur. J. of Opnl. Res.* 2, pp. 132-136.
- [51] LAMBRECHT M. and VANDER EECKEN J. and VANDERVECKEN H., (1981), Review of optimal and heuristic methods for a class of facilities in series dynamic lot size problems. *TIMS Studies in management sciences* , Vol. 16, North Holland Publishing Company, pp. 69-94.
- [52] LASDON L. S. and TERJUNG R. C., (1971), An efficient algorithm for the multi item scheduling. *Opns. Res.* 19, pp. 946-969.
- [53] LAWLER E.L., LENSTRA J.K, RINNOY KAN A.H.G, (1982), Recent developments in deterministic sequencing and scheduling: a survey. In *Deterministic and stochastic scheduling*, M.A.H. Dempster et al. (editors), Reidel publishing Company, pp. 35-73.
- [54] LOVE S. F., (1972), A facility in series inventory model with nested schedules. *Mgmt. Sc.* 28, pp. 327-338.

- [55] LOVE S. F., (1973), Bounded production and inventory models with piece-wise concave costs . *Mgmt. Sc.* 20, pp. 313-318.
- [56] LOVE R. R. Jr. and VEGUMANTI R. R. , (1978) The single plant mold allocation problem with capacity and changeover restrictionS. *Opns. Res.* 26, pp. 156-165.
- [57] MACNAUGHTON R., (1959), Scheduling with deadlines and loss functions. *Mgmt.Sc.* 6, pp. 1-12.
- [58] MANNE A. S., (1958), Programming of economic lot sizes. *Mgmt. Sc.* 4, pp. 115-135.
- [59] MARSTEN R. E., (1975), The use of the Boxstep method in discrete optimisation. In *Math. Programming Study 3, Non differentiable optimisation* , M. L. Balinski (ed), pp. 127-144.
- [60] MUCKSTAD J. and KOENIG S. A., (1977), An application of Lagrangean relaxation to scheduling in power generation systems. *Opns. Res.* 25, pp. 387-403.
- [61] MULLER MERBACH H., (1975), Modelling techniques and heuristics for combinatorial problems. In *Combinatorial programming: methods and applications*, B. Roy editor, D. Reidel Publishing Company, pp. 3-27.
- [62] NAUSS R., (1976), An efficient algorithm for the knapsack problem. *Mgmt. Sc.* 23, pp. 27-31.
- [63] PARKER R. G., DEANE R. H. and HOLMES R. A., (1977), On the use of a vehicle routing algorithm for the parallel processor problem with sequence dependent changeover costs. *AIIE Transactions* 9, pp. 155-160.
- [64] POLYAK B. T., (1967), A general method for solving extremum problems. *Soviet Math. Dokl.*, No 8, pp. 593-597.
- [65] PRABHAKHAR T., (1974), A production scheduling problem with sequencing considerations. *Mgmt. Sc.* 21, pp. 34-43.

- [66] RINNOY KAN A.H.G., LAGEWEL B.J., LENSTRA J.K. Minimising total costs in one machine scheduling. *Opns. Res.* 23, pp. 908-927.
- [67] ROSS G. T. SOLAND R. M., (1975), A branch and bound algorithm for the general assignment problem . *Math. Programming* 8, pp. 91-103.
- [68] SAHNEY V.K., (1972), Single server, two machines sequencing with switching time. *Opns. Res.* 20, pp. 24-36.
- [69] SHAPIRO J. F., (1979), A survey of Lagrangean techniques for discrete optimisation. In *Ann. of discrete mathematics* 5, North Holland publishing company, pp. 113-138.
- [70] SHOR N. Z., (1970), Utilisation of the space dilatation in the minimisation of convex functions. *Cybernetics* 6, pp. 7-15.
- [71] SMITH, (1956), Various optimisers for single stage production. *Naval Res. Logist. Quart.* 3, pp. 59-66.
- [72] STINSON, J.P., DAVIES E.W, KHUMAWALA B.M., (1978), Multiple resource constrained scheduling using branch and bounds. *A.I.I.E Transactions* 10, pp. 252-259.
- [73] SWOVELAND G., (1975), A deterministic multi-period production planning model with piecewise concave production and holding backorder costs. *Mgmt. Sc.* 21, pp. 1007-1013.
- [74] TAHA H. A., (1975), *Integer programming Theory application and computations* , Academic press.
- [75] TARJAN R. E., (1977), Finding optimum branchings. *Networks* 7, pp. 25-35.
- [76] WAGNER H. M. and WHITIN T. M., (1958), Dynamic version of the economic lot size model. *Mgmt. Sc.* 5, pp. 89-96.
- [77] YOUNG R. D., (1968), A simplified primal (all integer) integer programming algorithm. *Opns. Res.* 16, pp. 750-782.

- [78] ZANGWILL W. I., (1969), A backlogging model and a multi-echelon model of a dynamic economic lot size production system a network approach. *Mgmt. Sc.* 15, pp. 506-527.

Appendix A

The knapsack problem

For every packing line i , the following knapsack problem KP_i has to be solved:

$$\max \sum_j (\lambda_j - p_{i,j}) x_{i,j}$$

Subject to

$$\sum_j a_{i,j} x_{i,j} \leq T_i$$

It is well known [8] that when the variables are ordered such that the coefficients $(\lambda_j - p_{i,j})/a_{i,j}$ are in decreasing order, the solution to the linear program KPL_i obtained by dropping the integrality constraint is given by :

$$\begin{aligned} x_{i,j} &= 0 \quad \forall j \text{ such that } (\lambda_j - p_{i,j}) \leq 0 \\ x_{i,j} &= 1 \quad \forall j \mid (\lambda_j - p_{i,j})/a_{i,j} > (\lambda_r - p_{i,r})/a_{i,r} \\ x_{i,j} &= 0 \quad \forall j \mid (\lambda_j - p_{i,j})/a_{i,j} > (\lambda_r - p_{i,r})/a_{i,r} \\ x_{i,r} &= (T_i - \sum_{j,j \in J_1} a_{i,j})/a_{i,r} \end{aligned}$$

Where r is the last j such that

$$\sum_j a_{i,j} \geq T_i.$$

In other words, the items are loaded in the above defined order, until either there is overload or an exact fit, in both cases the last item loaded is r .

$$J_1 = \{j \mid x_{i,j} = 1\}$$

$$J_0 = \{j \mid x_{i,j} = 0\}$$

The value of the optimal solution to KPL_i is then

$$V(KPL_i) = \sum_{j \in J_1} p_{i,j} + p_{i,r} x_{i,r}$$

$V(KPL_i)$ thus gives an upper bound on $V(KP_i)$ which can be used in a branch and bound algorithm for solving KP_i . However one can do better. Ingargiola and Korsh [40] have shown that it is possible to reduce the dimension of the problem considerably, using inexpensive tests based on $V(KPL_i)$ which allow one to determine the values of a number of variables in the optimal solution of KP_i . Having determined these values, a process that was termed “pegging” in [62], it remains to find the values of the non-pegged variables using a tree search scheme, with LP relaxation.

R. Nauss proposed [62] a somewhat faster test based on a Lagrangean relaxation of KP_i .

The Lagrangean problem associated with KP_i is:

$$(KPG_i) \quad V(KPG)_i = \max \sum_j (\lambda_j - p_{i,j}) + \delta(T_i - \sum_j a_{i,j})$$

with $x_{i,j} = 0, 1$ The solution to KPG_i is known to be

$$\begin{aligned} x_{i,j} &= 1 && \text{if } \lambda_j - p_{i,j} - a_{i,j}\delta > 0 \\ x_{i,j} &= 0 && \text{if } \lambda_j - p_{i,j} - a_{i,j}\delta \leq 0 \end{aligned}$$

For the optimal multiplier δ

$$V(KPG_i) = V(KPL_i) \leq V(KP_i)$$

It is apparent that if in the optimal solution to $V(KPG_i)$, $x_{i,j} = 1$, then the solution to KPG_i with $x_{i,j} = 0$ will correspond to a decrease in $V(KPG)$ of $\lambda_j - p_{i,j} - a_{i,j}\delta$. In the same way, if in the optimal solution to KPG_i , $x_{i,j} = 0$, then the solution with $x_{i,j} = 1$ will correspond to an increase in $V(KPG_i)$ with the same value. Now let $x_{i,j} = 1$ in $V(KPL_i)$ and let a known solution to KP_i be $V(KP_i)^*$. From the above,

$$V(KPG_i)|_{x_{i,j}=0} = V(KPL_i) - (\lambda_j - p_{i,j} - a_{i,j}\delta)$$

It is easily seen that if

$$V(KPG_i)|_{x_{i,j}=0} \leq V(KP_i)^*$$

then $x_{i,j}$ must equal one in any optimal solution to KP_i .

Similarly, let $x_{i,j} = 0$ in $V(KPL_i)$. Then if

$$V(KPG_i)_{x_{i,j}=1} = v(KPL_i) + (\lambda_j - p_{i,j} - a_{i,j}\delta),$$

$x_{i,j}$ must be equal to zero in any optimal solution to KP .

The following branch and bound algorithm with reduction tests constructed by Nauss [62] has been adopted for solving KP_i .

Step1. $\forall j | \lambda_j - p_{i,j} \leq 0$ set $x_{i,j} = 0$ and define

$$J_t = \{j | \lambda_j - p_{i,j} > 0\}.$$

step2. Solve KPL_i for the variables in J_t to obtain an upper bound $V(KPL_i)$ and a fractional variable $x_{i,r}$. If $V(KPL_i)$ is feasible in problem KP_i then stop; otherwise

step2'. Compute a lower bound $V(KP_i)^*$ to KP_i , using a heuristic or

$$V(KP_i)^* = \sum_{j \in J_1} \lambda_j - p_{i,j}$$

where J_1 and J_0 are defined as above, and set the optimal multiplier $\delta^* = x_{i,r}/a_{i,r}$.

Initialise $I_1 = \{0\}$, $I_0 = \{0\}$

Step3. Pegging the variables:

$\forall j \in J_1$ if $V(KPL_i) - \lambda_j + p_{i,j} + a_{i,j}\delta^* \leq V(KP_i)^*$ then $x_{i,j}$ is pegged to one : $I_1 = I_1 \cup j$

$\forall j \in J_0$ if $V(KPL_i) + \lambda_j - p_{i,j} - a_{i,j}\delta^* \leq V(KP_i)^*$ then $x_{i,j}$ is pegged to zero : $I_0 = I_0 \cup j$

step3' Pegging by infeasibility:

$$\forall j \in J_0 \text{ if } a_{i,j} > T_i - \sum_{j \in J_1} a_{i,j} \text{ then } I_0 = I_0 \cup j$$

step4. If $J_i - J_1 \cup J_0 = 0$, then the optimum is found, stop; otherwise

Step5. Solve the reduced problem

$$(KPR_i) \quad \max \sum_{j,j \in J_i - J_1 \cup J_0} (\lambda_j - p_{i,j}) x_{i,j}$$

Subject to

$$\sum_{j,j \in J_i - J_1 \cup J_0} a_{i,j} x_{i,j} \leq T_i - \sum_{j \in J_1 \cup J_0} a_{i,j} \quad (\text{A.1})$$

using a branch and bound algorithm with LP relaxation and depth first search.

Appendix B

Algorithm for the shortest arborescence

Step0. Initial step

Step0a. Group all items being packed on all lines at the current node k in one component \mathcal{R} and let $\mathcal{X}(k)$ set of vertices consist of the items not yet loaded and $\mathcal{U}(k)$ the feasible changeovers.

Form the graph

$\mathcal{V}(k) = (\mathcal{R} \cup \mathcal{X}(k), \mathcal{U}(k))$ and the graph

$\mathcal{G}(k) = (\mathcal{X}(k), \mathcal{U}(k))$

Step0b. For every vertex l in $\mathcal{X}(k)$ select the arc (j_0, l) among all arcs (j, l) with $ct(j, l) < \infty$ such that

$ct(j_0, l) = \min_j ct(j, l)$

reduce the weight of all arcs (j, l) as follows

$ct^1(j, l) = ct(j, l) - ct(j_0, l) \quad \forall j$

store the selected arc in the list of arcs *LIST* and

update

$Y = Y + ct(j_0, l)$

set $p=1$

At the end of this process, a set of arcs \mathcal{W}^p is obtained.

Step1

Step1a If there are no cycles in \mathcal{W}^p then go to step 3,

otherwise

Step1b Contract every cycle in \mathcal{W}^p into a single new vertex, form a new directed graph $\mathcal{G}(k)^p$ consisting of the new vertices and the unchanged old. The arcs of $\mathcal{G}(k)^p$ have weight ct^p

step2. For every new vertex l in $\mathcal{G}(p)$ find j_0 such that

$$c^p(j_0, l) = \min_j c^p(j, l)$$

and reduce to obtain new weights:

$$c^{p+1}(j, l) = c^p(j, l) - c^p(j_0, l)$$

The new set of arcs \mathcal{W}^p is obtained, set $p = p + 1$ and goto step1

Step3. the weight of the shortest arborescence is Y .

The shortest arborescence is found by working backwards from $\mathcal{G}(p)$ and replacing each vertex created at p by the cycle it replaced. In each of these cycles, the arc that is directed toward the same vertex as one of the old arcs is deleted.

Appendix C

Data for packing lines problems

In this appendix and in the next, the following abbreviations will be used:

Bt : manufacturing unit processing is of the batch type

Con: manufacturing unit processing is of the continuous type

CNT: number of connections out of Silos (or material handling systems) to the packing lines.

Dd: Dedicated to a particular product.

D : demand.

Fl: Flexible silo.

h: hour.

H: Short term horizon.

MHS: Material handling system.

Mn: Men.

Prd: Product.

Pl: Packing line.

Sz: Size.

t: ton.

Table C.1: Parameters for problems series *P5*

Parameter	Value
Packing lines	5
Silos	7
CNT	1
Sizes	12
Products	7
Items	23
Operators	14
Fitters	10

Table C.2: Parameters for problems series *P6*

Parameter	Value
Packing lines	6
<i>MHS</i>	6
<i>CNT</i>	1
Products	7
Sizes	14
Items	25
Operators	24
Fitters	10

Table C.3: Data for problems series *P5*

<i>Item</i>	<i>Product</i>			<i>Size</i>	<i>Men</i>	<i>Lines</i>	<i>Demand in tons</i>	
	<i>P5₁</i>	<i>P5₂</i>	<i>P5₃</i>				<i>D1</i>	<i>D2</i>
1	1	1	1	1	3	1,2,3	140	160
2	2	2	2	1	3	1,2,3	140	140
3	3	3	3	1	3	1,2,3	100	100
4	4	4	4	1	3	1,2,3	100	150
5	1	1	1	2	3	1,2,3	140	110
6	2	2	2	2	3	1,2,3	110	140
7	6	3	3	2	3	1,2,3	110	110
8	1	1	1	3	3	1,2,3	65	70
9	2	2	2	4	3	1,2,3	82	50
10	6	6	6	5	3	1,2,3	82	60
11	5	5	5	6	3	1,2,3	100	80
12	7	2	2	6	3	1,2,3	60	75
13	2	2	2	7	3	1,2,3	50	40
14	7	7	7	7	3	1,2,3	50	45
15	1	1	1	8	3	4,5	70	70
16	4	1	1	9	3	4,5	60	60
17	1	1	1	10	3	4,5	40	34
18	3	3	3	10	3	4,5	45	48
19	2	2	2	11	4	4,5	34	58
20	3	4	4	11	3	4,5	90	60
21	1	1	1	12	4	4,5	10	15
22	3	3	3	12	4	4,5	32	32
23	7	7	4	12	3	4,5	20	20

Table C.4: Data for problems series *P6*

<i>Item</i>	<i>Prd</i>	<i>Sz</i>	<i>Mn</i>	<i>PL</i>	<i>Demand</i>				
					<i>D1</i>	<i>D2</i>	<i>D3</i>	<i>D4</i>	<i>D5</i>
1	1	1	7	1,2,3	250	200	150	220	150
2	2	1	7	1,2,3	150	150	110	110	130
3	3	2	7	1,2,3	150	150	80	140	80
4	4	2	5	1,2,3	130	130	80	100	80
5	5	3	6	1,2,3	130	130	80	80	100
6	6	4	6	1,2,3	150	120	90	90	90
7	7	4	6	1,2,3	120	150	150	120	150
8	3	5	6	1,2,3	190	150	100	80	100
9	4	5	5	1,2,3	190	130	110	130	110
10	1	6	6	1,2,3	90	90	140	110	130
11	2	7	5	1,2,3	70	120	90	70	90
12	5	7	5	1,2,3	70	80	110	80	100
13	6	7	3	1,2,3	45	45	100	55	100
14	3	8	3	1,2,3	25	29	46	49	25
15	7	8	3	4,5	30	35	20	35	45
16	2	9	3	4,5	25	25	30	25	30
17	5	10	3	4,5	45	45	45	30	45
18	6	10	3	4,5	22	22	14	14	14
19	1	11	3	4,5	14	18	10	18	10
20	4	11	3	4,5	24	24	16	16	15
21	3	12	6	6	2.3	2.3	7	5.3	4.3
22	1	13	5	6	6.4	4.4	9	6.4	4
23	4	13	6	6	4.8	4.8	4.8	4.8	3
24	2	14	6	6	3.	6.	7	3	5
25	7	14	6	6	3.	3.	3	3	3

Table C.5: parameters for problems series *P71*

Parameter	Value
Packing lines	7
<i>MHS</i>	3
<i>CNT</i>	2
Products	8
Sizes	15
Items	30
Operators	25
Fitters	10

Table C.6: Parameters for problems series *P72*

Parameter	Value
Packing lines	7
<i>MHS</i>	4
<i>CNT</i>	2
Products	10
Sizes	16
Items	30
Operators	28
Fitters	12

Table C.7: Data for problems series *P71*

<i>Item</i>	<i>Prd</i>	<i>Sz</i>	<i>Mn</i>	<i>PL</i>	<i>demand</i>			
					<i>D1</i>	<i>D2</i>	<i>D3</i>	<i>D4</i>
1	1	1	7	1,2,3	320	280	200	320
2	2	1	7	1,2,3	150	150	200	150
3	3	1	9	1,2,3	150	170	80	170
4	6	1	5	1,2,3	150	150	200	80
5	4	2	5	1,2,3	130	100	80	80
6	5	2	6	1,2,3	130	130	100	130
7	7	2	6	1,2,3	130	140	80	80
8	8	2	6	1,2,3	130	110	70	90
9	3	3	5	1,2,3	70	80	80	120
10	5	3	5	1,2,3	70	50	200	120
11	1	4	3	1,2,3	60	60	60	90
12	2	5	3	1,2,3	120	60	60	120
13	6	5	3	1,2,3	64	80	80	80
14	7	5	3	1,2,3	50	70	40	50
15	8	6	3	1,2,3	70	70	60	100
16	4	7	3	4,5	55	65	90	78
17	3	8	3	4,5	70	70	50	50
18	7	8	3	4,5	40	60	40	40
19	1	9	3	4,5	60	60	60	90
20	6	10	3	4,5	70	70	100	100
21	5	11	3	4,5	50	50	70	50
22	2	12	7	6,7	16	20	18	12
23	3	12	7	6,7	15	25	20	15
24	1	13	7	6,7	14	19	14	14
25	2	13	6	6,7	15	18	18	10
26	4	14	6	6,7	3	3	3	5
27	5	14	7	6,7	3.4	4.4	4.4	5.4
28	6	15	6	6,7	3.4	5.4	5.4	6.4
29	7	15	6	6,7	7	4	6	3
30	8	15	6	6,7	5	4	5	3

Table C.8: Data for problem series *P72*

<i>Item</i>	<i>Prd</i>	<i>Sz</i>	<i>Mn</i>	<i>PL</i>	<i>demand</i>				
					<i>D1</i>	<i>D2</i>	<i>D3</i>	<i>D4</i>	<i>D5</i>
1	1	1	7	1,2,3,4	260	260	160	260	160
2	2	1	7	1,2,3,4	100	200	150	200	200
3	3	1	7	1,2,3,4	100	130	200	130	200
4	6	2	7	1,2,3,4	100	130	90	130	150
5	9	2	7	1,2,3,4	150	150	220	120	220
6	1	2	9	1,2,3,4	90	120	150	140	120
7	4	3	7	1,2,3,4	90	90	200	110	200
8	7	3	7	1,2,3,4	80	100	90	100	50
9	8	3	7	1,2,3,4	90	90	90	50	50
10	10	4	8	1,2,3,4	100	90	100	100	80
11	5	4	7	1,2,3,4	60	100	100	50	100
12	3	5	7	1,2,3,4	90	90	150	150	90
13	1	6	5	1,2,3,4	40	80	80	100	100
14	6	6	6	1,2,3,4	40	70	50	50	80
15	2	7	6	1,2,3,4	80	80	80	80	80
16	4	7	5	1,2,3,4	50	50	50	50	40
17	7	8	7	1,2,3,4	60	60	30	30	90
18	8	9	8	1,2,3,4	30	55	30	50	80
19	9	9	5	1,2,3,4	50	80	40	60	67
20	1	10	7	1,2,3,4	45	45	30	40	45
21	10	10	4	1,2,3,4	45	45	45	45	60
22	3	11	5	1,2,3,4	30	50	40	40	50
23	6	11	4	1,2,3,4	32	42	30	50	40
24	5	12	4	1,2,3,4	24	24	45	24	35
25	1	13	4	1,2,3,4	30	35	26	30	40
28	3	14	3	1,2,3,4	20	20	30	23	35
27	4	14	5	1,2,3,4	15	20	25	20	20
28	7	15	3	1,2,3,4	3	3	8	4	5
29	10	16	3	1,2,3,4	5	3	6	6	6
30	2	16	3	1,2,3,4	7	5	9	4	2

Appendix D

Data for two-stage problems

Table D.1: Parameters for problem *PSP1*

Parameter	value
Manufacturing units	1
Packing lines	6
Silos	7
Material handling systems	2
Sizes	5
Products	4
Items	15
Operators	35
Fitters	10

Table D.2: Manufacturing unit characteristics for *PSP1*

Manufacturing units charact.					
Unit	Type	Max rates in tons per hour			
		<i>Prd1</i>	<i>Prd2</i>	<i>Prd3</i>	<i>Prd4</i>
1	<i>Con.</i>	26	26	24	24

Table D.3: Data for problem *PSP1*

<i>Item</i>		<i>Mn</i>	<i>Pl</i>	<i>Rt</i>	<i>H1</i>		<i>H2</i>		<i>H3</i>		<i>H4</i>	
<i>Prd</i>	<i>Sz</i>				<i>D</i>	<i>Sbl</i>	<i>D</i>	<i>Sbl</i>	<i>D</i>	<i>Sbl</i>	<i>D</i>	<i>Sbl</i>
1	1	3	2	1.74	27	1	45	1	0		36	1
1	2	5	1,3	6.7	145	2	80	1	125	2	100	1
1	4	7	4,5	12.2	290	2	250	2	430	3	330	3
1	5	7	6	12.2	180	1	180	1	70	1	80	1
2	1	3	2	1.8	36	1	0		21	1	0	
2	2	5	1,3	7	70	1	90	1	70	1	80	1
2	4	7	4,5	12.7	290	2	360	3	400	3	250	2
2	5	7	6	12.7	90	1	200	2	0		110	1
3	1	3	2		1.6	18	1	16	1	0	17	1
3	2	5	1,3	6.2	150	1	40	1	215	2	75	1
3	4	7	4,5	11.2	725	4	475	3	475	3	525	4
3	5	7	6	11.2	200	2	125	1	270	2 00	2	
4	1	3	2	2	11	1	0		18 1		0	
4	2	5	1,3	7.4	55	1	70	1	125	1	125	1
4	3	7	5,6	9.3	145	1	160	2	170	1	135	1

Table D.4: Silos characteristics for *PSP1*

Silos charact.					
Silo	Type	Capacity in tons			
		<i>Prd1</i>	<i>Prd2</i>	<i>Prd3</i>	<i>Prd4</i>
1,2,3,4	<i>Dd</i>	25.5	25.5	22.5	22.5
5,6,7	<i>Fl</i>	46.5	46.5	46.5	46.5

Table D.5: Parameters for problem *PSP2*

Parameter	Value
Manufacturing units	4
Packing lines	4
Silos	7
Material handling systems	0
Sizes	6
Products	7
Items	18
Operators	10
Fitters	10

Table D.6: Manufacturing units characteristics for *PSP2*

Manufacturing units charact.								
Unit	Type	Rates						
		<i>Prd1</i>	<i>Prd2</i>	<i>Prd3</i>	<i>Prd4</i>	<i>Prd5</i>	<i>Prd6</i>	<i>Prd7</i>
1	<i>Bt.</i>	10t/2.4 h						
2	<i>Bt</i>	10t/2.4 h	10t/1.5h	10t/1.5h	10t/1.5h	10t/1.5h		
3	<i>Bt</i>						10t/1.5h	
4	<i>Bt</i>							5t/2h

Table D.7: Silos characteristics for problem *PSP2*

Silos charact.								
Silo	Type	Capacity in tons						
1,2,3,4,5,6,7	<i>Dd</i>	<i>Prd1</i>	<i>Prd2</i>	<i>Prd3</i>	<i>Prd4</i>	<i>P5</i>	<i>P6</i>	<i>Prd7</i>
		35t	62	32	32	31	45	29

Table D.8: Data for problem *PSP2*

<i>Item</i>		<i>Mn</i>	<i>Pl</i>	<i>Rt</i>	<i>H1</i>		<i>H2</i>		<i>H3</i>		<i>H4</i>	
<i>Prd</i>	<i>Sz</i>				<i>D</i>	<i>Sbl</i>	<i>D</i>	<i>Sbl</i>	<i>D</i>	<i>Sbl</i>	<i>D</i>	<i>Sbl</i>
1	3	3	1	3.75	45	1	40	1	0		0	
1	5	3	2,4	5	150	1	150	1	150	1	150	1
1	6	3	2,4	5	80	1	0		60	1	60	1
2	2	3	1	3.25	63	1	0		63	1	0	
2	3	3	1	3.75	30	1	30	1	30	1	30	1
2	4	3	1,2,4	4.375	70	1	70	1	70	1	70	1
2	6	3	2,4	5	15	1	15	1	15	1	15	1
3	3	3	1	3.75	68	1	0		60	1		
3	5	3	2,4	5	45	1	40	1	0	1	0	
4	3	3	1	3.75	30	1	0	1	45	1	0	
4	5	3	2,4	5	0	1	0	1	0	1	0	
5	3	3	1	3.75	20	1	14	1	51	1	0	
5	5	3	2,4	5.625	45	1	40	1	0	1	0	
6	3	3	1	3.75	60	1	0		90	1	0	
6	5	3	2,4	5.625	60	1	60	1	60	1	60	1
6	6	3	2,4	5	200	1	0		0		0	
7	1	3	3	1.5	70	1	70	1	70	1	70	1
7	3	3	3	1.875	90	1	0		135	1	0	

ProQuest Number: 29205943

INFORMATION TO ALL USERS

The quality and completeness of this reproduction is dependent on the quality and completeness of the copy made available to ProQuest.



Distributed by ProQuest LLC (2022).

Copyright of the Dissertation is held by the Author unless otherwise noted.

This work may be used in accordance with the terms of the Creative Commons license or other rights statement, as indicated in the copyright statement or in the metadata associated with this work. Unless otherwise specified in the copyright statement or the metadata, all rights are reserved by the copyright holder.

This work is protected against unauthorized copying under Title 17,
United States Code and other applicable copyright laws.

Microform Edition where available © ProQuest LLC. No reproduction or digitization of the Microform Edition is authorized without permission of ProQuest LLC.

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 - 1346 USA