

FOR REFERENCE

NOT TO BE TAKEN FROM THIS ROOM

MODELS FOR A MULTI-ITEM
PRODUCTION SYSTEM

by

SEÇKİN İKİZ

B.S. in M.E., Boğaziçi University, 1983

Bogazici University Library



39001100314544

14

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of

Master of Science

in

Industrial Engineering

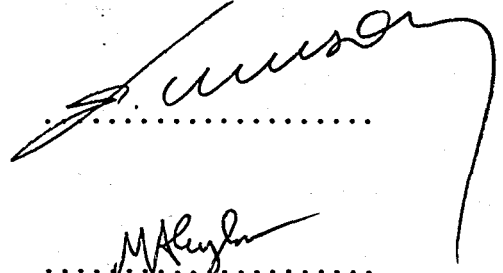
Boğaziçi University

1985

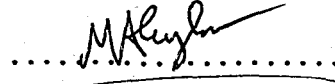
MODELS FOR A MULTI-ITEM
PRODUCTION SYSTEM

APPROVED BY

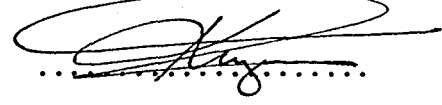
Doç. Dr. Gündüz ULUSOY
(Thesis Supervisor)



Doç. Dr. M. Akif EYLER



Doç. Dr. Ahmet KUZUCU



DATE OF APPROVAL

182938



ACKNOWLEDGEMENTS

I want to express my gratitude to Doç.Dr. Gündüz Ulusoy for his generous contribution and infallible guidance throughout this study.

I also thank Doç.Dr. Akif Eyler and Doç.Dr. Ahmet Kuzucu for their interest and evaluation as members of the thesis committee.

I owe much to Zehra Eliçin and Ramazan Uzun from PİMAŞ A.Ş. whose valuable suggestions led to more accurate modelling of the problems handled in this study.

At last I want to thank Alper Oysal for the painstaking typing.

MODELS FOR A MULTI-ITEM PRODUCTION SYSTEM

A B S T R A C T

To minimize work-in-process, or equivalently mean flow time of capital has been an emphatic objective in production. It deserves still greater attention in today's Turkish Economy governed by monetaristic austerity measures.

This study proposes models for two problems of a multi-item production system.

One of them is an algorithm organizing activities at three levels of the production process. The second one is an implicit enumeration algorithm rendering selection of orders for simultaneous release when the current inventory is incapable of satisfying all of them. Both are designed so as to serve the aim of minimizing mean flow time of capital bound to the process.

These models can be implemented separately or in combination depending on the inventory/sales policy of the application.

COKÜRÜNLÜ BİR ÜRETİM SİSTEMİ İÇİN MODELLER

Ö Z E T

Süreçteki iş miktarının ya da kapitalin süreçte tutulduğu ortalama sürenin enazlanması üretimde üzerinde önemle durulan bir amaç olagelmıştır. Monetarist istikrar önlemleri ile yönetilen günümüz Türk Ekonomisinde bu amaç daha da fazla önem kazanmaktadır.

Bu çalışma çokürünlü bir üretim sisteminin iki problemi için modeller önermektedir.

Bunlardan biri üretim sürecinin üç düzeyindeki etkinlikleri düzenleyen bir algoritmadır. İkincisi ise mevcut envanterin tüm siparişleri karşılamaya yeterli olmadığı durumlarda hangilerinin anında karşılanacağına karar veren bir örtük birerleme (implicit enumeration) algoritmasıdır. Her iki model de kapitalin süreçte tutulduğu ortalama süreyi enazlama amacına hizmet edecek şekilde tasarlanmıştır.

Bu modeller envanter/satış politikasına bağlı olarak tek başına veya birlikte uygulanabilirler.

TABLE OF CONTENTS

	<u>Page</u>
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZET	v
LIST OF FIGURES	viii
I. INTRODUCTION	1
II. A DYNAMIC HIERARCHICAL APPROACH TO A MULTI-LEVEL SCHEDULING PROBLEM	4
2.1 The Problem	4
2.2 A Constructive Algorithm; FCR	7
2.3 A Heuristic Algorithm for the Multi-Level Scheduling Problem	12
2.3.1 The Reasons for the Hierarchical Approach	12
2.3.2 Assumptions	15
2.3.3 The Algorithm	17
2.3.3.1 Initialization	17
2.3.3.2 Phase One	17
2.3.3.3 Phase Two	22
2.4 Tests on the Algorithm	24
2.4.1 Factors	24
2.4.2 Tests and Results	26
2.4.3 Observations	27
2.5 Implementation of the Algorithm	28
2.6 Possible Extensions	29

	<u>Page</u>
III. AN IMPLICIT ENUMERATION ALGORITHM FOR THE ORDER SELECTION PROBLEM	31
3.1 The Problem	31
3.2 An Implicit Enumeration Algorithm	33
3.3 The Proposed Heuristics for the Selection of the Partitioning Variable	36
3.4 Results	38
IV. CONCLUSION	44
REFERENCES	47
APPENDIX	48

LIST OF FIGURES

	<u>Page</u>
FIGURE 2.1.1 Armature Production	5
FIGURE 2.2.1 Comparison of S and S'	9
FIGURE 2.3.1 Three Levels of the Problem	14
FIGURE 2.3.2 Flowchart for the 3-level Scheduling Algorithm	18
FIGURE 2.3.3 Flowchart for the Mold/Machine Scheduling Algorithm for a Particular Assembly Job	21
FIGURE 3.2.1 Flowchart for the Implicit Enumeration Algorithm	35
FIGURE 3.4.1 Results for Case A	40
FIGURE 3.4.2 Results for Case B	41
FIGURE 3.4.3 Results for Case C	42

I. INTRODUCTION

This thesis involves study on two main topics which are interrelated in the context of a real life problem.

One of them is a scheduling problem of a special nature. Encountered in production of plastic armatures, a process involving three levels of scheduling decisions, it can be defined as a multi-level scheduling problem. The so called lowest level decisions determine the schedule of molds on injection molding machines. Each mold can be mounted on certain machines. Plastic components of armatures are the outputs of the molding process which feed the assembly shop for the assembly of end-products. The mid-level decision is scheduling the assembly of different types of armatures. The third and the highest level is the sequencing of order releases (or sales) which are made up of selection of end-products at various quantities.

Coming from the wholesalers of the firm's products, the orders, in general, don't have strict due dates. However, the earlier an order is dispatched, the earlier is the receipt of its payment. Thus, it is important for the firm to make

ready the end-product requirements of orders as early as possible. In other words, it is desirable to minimize the delay of revenue receipts, or equivalently the mean flow time of capital in process.

The company holds stocks of armatures as well as the constituent parts. Thus, some of the periodic orders can be met immediately before starting up with production. The problem of selecting among all orders those to be answered immediately is the concern of the second part of the thesis.

The above rationale is valid in this problem, too. The aim is to minimize the average delay of potential revenues. Thus, the total monetary worth of orders selected for instant satisfaction should be maximized subject to the constraint of current inventory levels.

Austerity measures being practiced by the government force the business enterprises to work with extensive capital. Decision making in production, thus, have to provide ways of speeding up the flow of capital.

The theme of this study have thus been inspired by the facts of the Turkish Economy in the 1980's.

The second chapter gives an account of the procedure developed for the first problem. A polynomial time algorithm, named FCR (Fastest Capital Release), minimizing mean-flow time of capital for a "single machine" scheduling problem is introduced. It provides the framework for the overall 3-level scheduling algorithm which covers the rest of the chapter.

Chapter Three is on the order selection problem. A 0/1 programming formulation (recognized as a 0/1 multidimensional knapsack problem) is constructed, and an implicit enumeration algorithm exploiting the nature of the problem is developed.

Eleven decision rules are proposed for the selection of the partitioning variable. Finally, performances of these rules are compared with varying; (a) number of orders, (b) number of products, and (c) levels of inventory. To note, a rule involving minimum of computational burden indisputably comes out to be the most effective one.

The last chapter concludes the thesis by suggesting ways the proposed methods can be used separately or in combination under various operating conditions.

II. A DYNAMIC HIERARCHICAL APPROACH TO A MULTI-LEVEL SCHEDULING PROBLEM

2.1. The Problem

A company well known as a producer of plastic construction materials, a year ago began producing plastic armatures. Nine different types, each with four color options were made available. Of the parts making up the end-products, plastic ones are manufactured by the firm, metal and gasket parts are supplied from outside vendors.

Plastic components are produced by injection molding of granulated thermoplastics.

44 different molds can be mounted on one, two, or three of four available molding machines. About two thirds of plastic components are common to two or more types of armatures. Again, about two thirds are colored components.

The process is summarized in Figure 2.1.1.

Production engineers had difficulty in synchronizing molding and assembly processes mainly due to

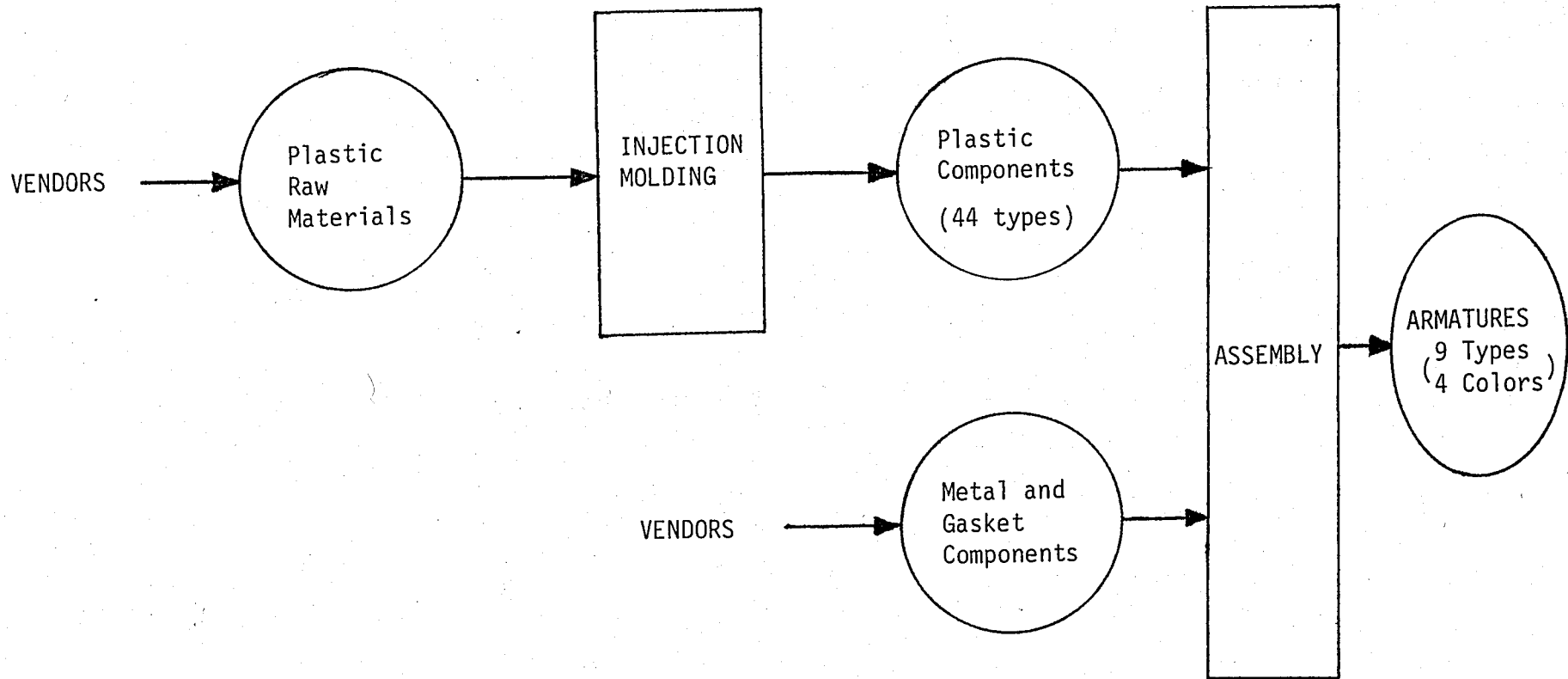


Figure 2.1.1 Armature Production

- (A) intricate bills of materials for end-products, and
- (B) mold/machine assignment restrictions.

Lack of systematic programming for production was causing long waiting times of finished plastic components before the assembly process since assembly of an end-product could not start before all of its components were ready. A similar trouble was felt at the sales end. Sales were made to retailers, and thus, orders demanded various quantities of almost all product types. This time, finished end-products had to wait long prior to sale for the missing types to be assembled. Customers being the retailers of the firm's products, orders didn't have strict due dates, but the earlier an order was sent, the earlier was the receipt of its payment.

Through demand for products was being answered promptly, it was achieved at the expense of holding a large working capital in the form of finished products and components.

It was expected, after these observations, that a systematic mold/machine scheduling in connection with the assembly scheduling of armature types would reduce the in-between waiting times of plastic components. Furthermore, before-sales waiting times of finished products could be reduced by taking into account orders' constituent product demands while fixing the assembly schedule. Thus, on the overall, flow of capital bound to the process could be hastened significantly.

As could have been felt already, this study emphasizes the rate at which money is used to gain returns. In view of high interest rates and the rising competitiveness in today's

Turkish Economy, this is a critical measure of effectiveness for a business enterprise. The longer a quantity of money sojourns in a process, the greater will be its cost. Therefore it must be an eligible approach for any entrepreneurial activity to minimize the mean duration a unit of capital is in the process.

The following section introduces an n jobs/single machine scheduling rule which minimizes the mean flow time of capital contained in these jobs. This "constructive algorithm" will be the basis of the general algorithm for the solution of the above posed problem.

2.2 A Constructive Algorithm, FCR

Fastest Capital Release (or FCR) Algorithm is a constructive algorithm; one, as defined by French [1], "... which builds up an optimal solution from the data of the problem by following a simple set of rules which exactly determine the processing order."

Let us define the considered problem :

There are n types of products to be processed on a single machine. The following data are available;

- x_i , $i = 1, \dots, n$: number of to-be-processed products of type i ,
- c_i , $i = 1, \dots, n$: amount of capital contained in one unit of product of type i ,
- p_i , $i = 1, \dots, n$: processing time required by one unit of product of type i .

The problem is to determine the sequence of jobs which renders the mean flow time of capital bound to process be minimum.

The rule for the solution of this problem is stated below.

THEOREM : (The Fastest Capital Release (FCR) rule)

For the problem defined above, mean flow time of capital is minimized by sequencing the lumped-jobs of types $i = 1, \dots, n$ such that

$$\frac{c_{i(1)}}{p_{i(1)}} \geq \frac{c_{i(2)}}{p_{i(2)}} \geq \dots \geq \frac{c_{i(n)}}{p_{i(n)}}$$

where $i(k)$ denotes the job that is processed k th.

Proof. Let S and S' be two schedules differing only in the order of two lumped-jobs 1 and 2 with distinct (c/p) ratios (see Fig. 2.2.1a). The cumulative released capital versus time profiles will differ only for the time interval during which jobs 1 and 2 are processed (see Fig. 2.2.1b).

The profile for schedule S follows the dashed line whereas that for S' follows the solid line from A to E.

The area above the cumulative released capital versus time profile is held capital-time which is a measure of liability for the schedule. The mean flow time of capital for a certain schedule is the ratio of total held capital-time it incurs to total capital it involves.

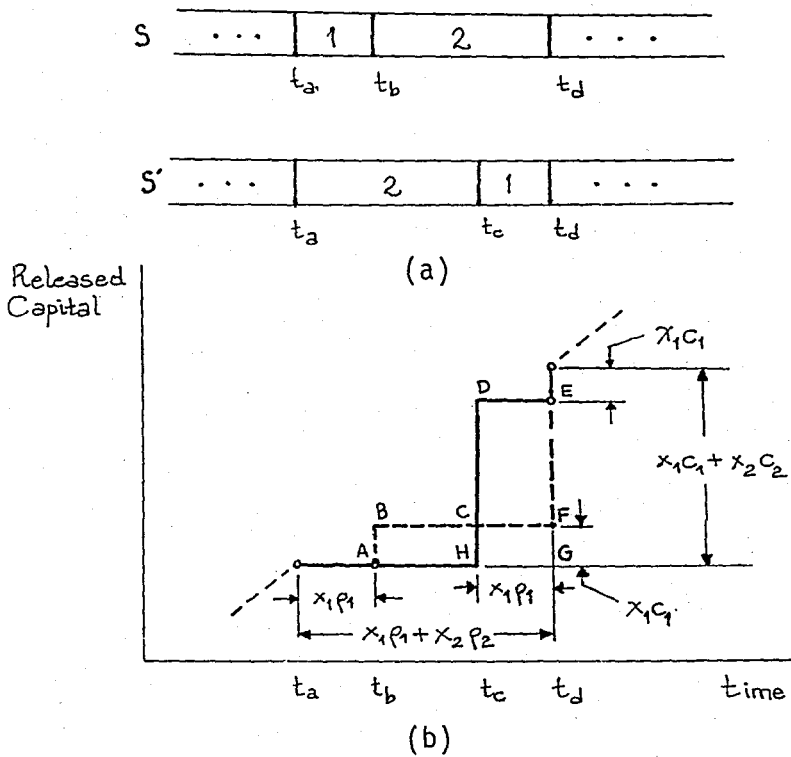


Figure 2.2.1 Comparison of S and S'

Since total capital is a constant, mean flow time is proportional to the area above the profile. Returning to Fig. 2.2.1, performance comparison for schedules S and S' can thus be made solely upon comparison of areas ABCFGHA and DEFGHCD. If the first area is larger than the second, schedule S is superior to schedule S', and vice versa.

Let us compare the two areas:

$$\frac{S(\text{ABCFGHA})}{S(\text{DEFGHCD})} = \frac{(x_1 c_1)(x_2 p_2)}{(x_2 c_2)(x_1 p_1)} = \frac{c_1 p_2}{c_2 p_1} = \frac{(c_1/p_1)}{(c_2/p_2)}$$

Thus, if $\frac{c_1}{p_1} > \frac{c_2}{p_2}$ then S is superior to S', and

if $\frac{c_2}{p_2} > \frac{c_1}{p_1}$ then S' is superior to S.

Therefore it can be concluded that while considering two neighbor jobs, ordering them in decreasing (c/p) decreases the mean flow time of capital for the overall process. If the ratios are equal, their ordering is ineffective in this concern.

This rule can be induced for the overall schedule, simply by figuring out that two-by-two comparisons of (c/p) ratios of n jobs should eventually lead to the optimal schedule. \square

Two-by-two comparisons of n entities require at most

$$\binom{n}{2} = \frac{n!}{(n-2)!2!} = \frac{n(n-1)}{2} = \frac{1}{2} n^2 - \frac{1}{2} n$$

calculations. Thus, number of computations to construct the optimal schedule grows polynomially with the size of the problem. Thus, as all other constructive algorithms, FCR algorithm too, belongs to P class of time complexity.

There is a remarkable semblance between the FCR algorithm and the Shortest Processing Time (or SPT) algorithm. SPT, as defined in Conway et al. [2] minimizes the mean flow time of jobs in an n jobs/single machine schedule by sequencing the jobs in non-increasing processing times. FCR, differing from SPT, decomposes various jobs into identical value units and minimizes their mean flow time. It is apparent that FCR, with respect to SPT, provides a more flexible method to minimize the flow rate of material (or work)-in-process.

Above, "recovered working capital" was considered as a measure of returns for the production process. An alternative which is widely respected in OR literature is "gained profit".

May profit alone be a measure of returns for capital used in a process? In consideration of time value of money, it may not turn out so. As much as the profit gained from it, the size of the working capital becomes important in this respect. An eligible measure might be the ratio of profit to the capital bringing that profit. Still another one may be the revenue; capital plus profit, to break ties when profit ratios of jobs are not significantly different. The measure adopted in the argument above, namely capital-in-process, also assumes the case when the relative profit is invariant among jobs.

Two more points about Theorem deserve mentioning. Above, identical jobs of a certain type were considered as a whole and it was named a "lumped-job". Consequently, all jobs comprising a lumped-job were assumed incomplete before time enough to cover their total processing times elapses. In general, this may not be the case. Then, if all jobs are treated individually, non-decreasing (c/p) sequence would cluster like-type jobs at the optimal schedule. Thus, our simplifying assumption was made without loss of generality. Discussion above also makes clear that no improvement may be gained in the optimal schedule by allowing pre-emption.

The last word is for the set-up times. In practice, set-up times required prior to processing a batch of jobs may be significant. This case may be accommodated by FCR

algorithm via replacing (c_i/p_i) by

$$\frac{c_i}{p_i + (s_i/x_i)}$$

where s_i is the sequence-independent set-up time per batch of type i . With this modification, lumped-jobs schedule is still guaranteed to be optimal. Here, it is certain that pre-emption deteriorates the optimal solution.

2.3 A Heuristic Algorithm for the Multi-level Scheduling Problem

This section presents an algorithm proposed for the solution of the special scheduling problem of section 1.1.

2.3.1 The Reasons for the Hierarchical Approach

As discussed above, this problem is characterized by its multi-level structure. An intuitively appealing description for the anatomy of the problem is given below.

LEVEL 1 :

Scheduling n non-identical molds on m non-identical machines.

LEVEL 2 :

Scheduling p assembly jobs in series in an assembly shop. Outputs of the molds at level 1 impose not-earlier-than restraints on assembly start times.

LEVEL 3 :

Sequencing q order releases. Assembly completion times of end-products at level 2 impose not-earlier-than restraints on release times.

OBJECTIVE :

Synchronizing the activities at 3 levels so as to minimize the mean sojourn time of capital from the time injection molding starts to the moment it is released to meet an order.

Figure 2.3.1 depicts the 3-level structure of the problem.

The multi-echelon structure suggests a hierarchical approach. More explicitly, a proper way to deal with the problem is decomposing it into its levels and treating each sub-problem individually without losing sight of (a) the possible interactions between levels, and (b) the overall objective.

Such a preference is totally heuristic and thus liable to produce sub-optimal solutions. Several authors, to mention, Müller Merbach [3] and Silver et al. [4] justify decomposition methods mainly due to their capacity for conforming to the structure of the problem environment.

The alternative is constructing a single model and determining all scheduling decisions in one pass.

Besides causing loss of view of the functional interrelationships between activities, such an approach will

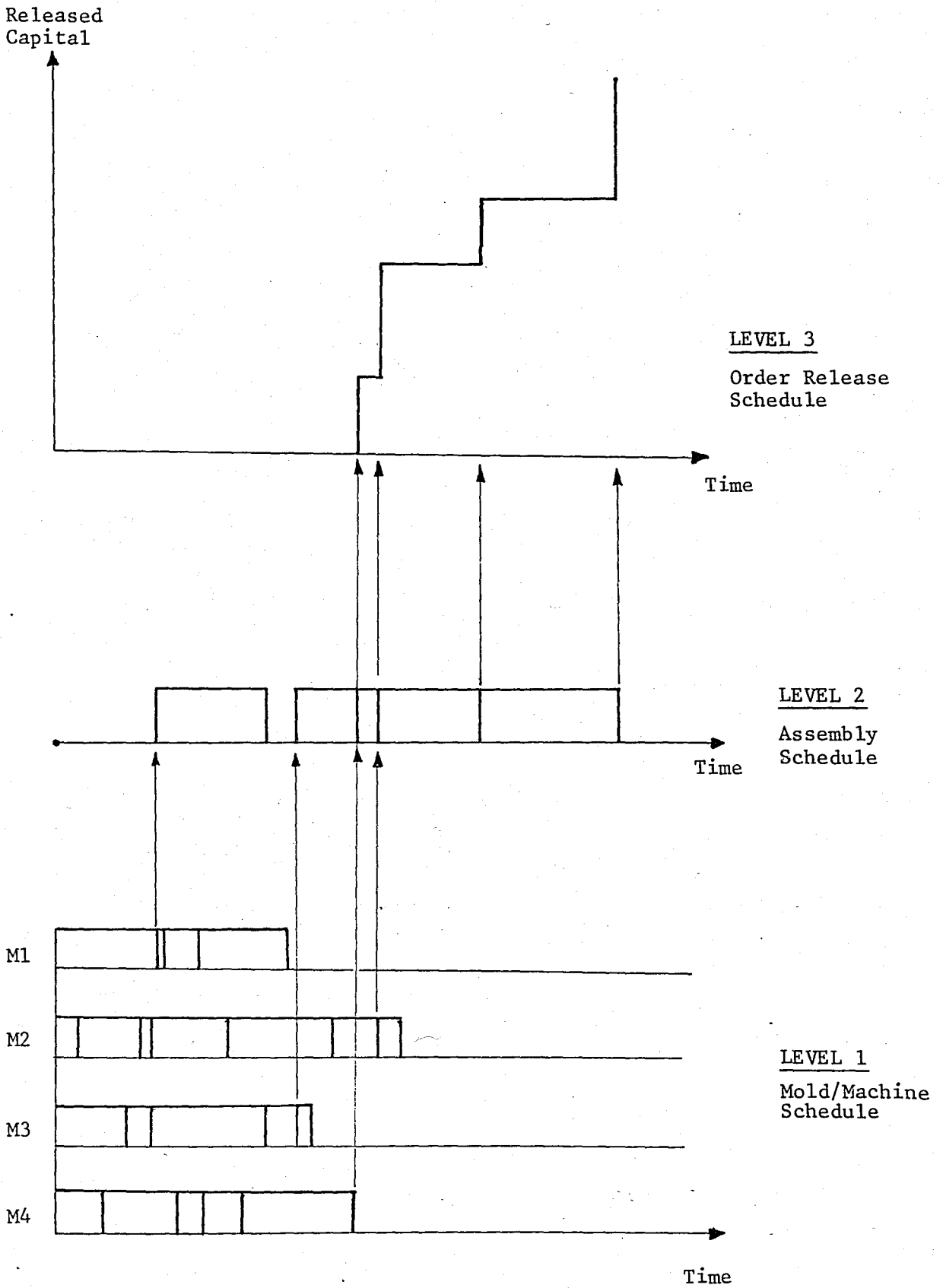


Figure 2.3.1 Three Levels of the Problem

prove unpractical by requiring a huge computational capacity due to combinatorial nature of the problem.

2.3.2 Assumptions

The underlying assumptions for the proposed algorithm are;

- (i) no pre-emption is allowed for molding jobs;
- (ii) no pre-emption is allowed for assembly jobs;
- (iii) an assembly job can start only after all molding jobs associated with its constituent parts are complete;
- (iv) an order can be released after the last assembly job it is waiting for completion produces enough to satisfy its demand for the associated end-product;
- (v) defective injection output is detected during production, and injection standard times are inflated to include the average delays that occur thereby;
- (vi) molding job for a certain plastic component is made up of lots of different colors. No set-up time is required for color shifts;
- (vii) assembly job for a certain armature is made up of lots of different colors. No set-up time is required for color shifts.

The first assumption is justified by the damage risk of molds which have no spares. Molding jobs have given economical lot sizes arising from the trade-off between the between the expected damage cost and the holding cost for extra production (see Appendix).

One reason behind the second assumption was discussed at section 2.2; pre-emption incurs extra set-up time. Another is due to need for simplifying the problem.

The third assumption may seem arbitrary in view of the fact that an assembly job may start at the moment the last molding jobs it waits for, produce just as much as it will use. The assumption, however serves for simplification of both the algorithm and the flow of finished components. Another point is that additional waiting caused becomes fairly short due to the nature of mold scheduling algorithm which assigns jobs on machines in decreasing order of processing times.

The fifth assumption reflects a fact of the problem environment. Quality control is performed by the operators and defective output rate is fairly low due to precise technology.

The sixth assumption is made considering that shift from one color to another does not require mold adjustment but only a change of the raw-material container, and that mixed-color scrap output takes negligible time.

The last assumption neglects the time spent during preparation of colored components prior to shift to another color for the same type of armature.

An assumption not appearing in the above list have been accepted while defining the problem; assembly jobs would be precessed in series. It was imposed by the restricted space available for assembly. Besides it became very useful for the simplification of the problem.

2.3.3 The Algorithm

The algorithm rendering a low mean flow time of capital for the 3-level scheduling problem is depicted in Figure 2.3.2.

2.3.3.1 Initialization

The First Step involves preliminary calculations. Given, order descriptions (i.e. selections among available type and color of armatures), total demand for end-products and consequently total requirements for plastic components are determined. Those components which have insufficient stocks are accepted as molding jobs and their production quantities are set to the greater of the economical lot size and the amount required for assembly.¹

The rest of the algorithm can be considered as two subsequent phases. These two phases suggest two different hierarchical frameworks.

2.3.3.2 Phase One

First, unscheduled assembly jobs are considered one by one, and for each, a mold/machine schedule providing earliest start for assembly is constructed. Among these candidate schedules, the one which enables an assembly job start with minimum delay after the last scheduled assembly job, is picked and adopted. The associated assembly job is

¹ Economical lot size is by definition a multiple of average periodic demand; thus it is an exceptional case when the immediate requirement for a period exceeds it (See Appendix)

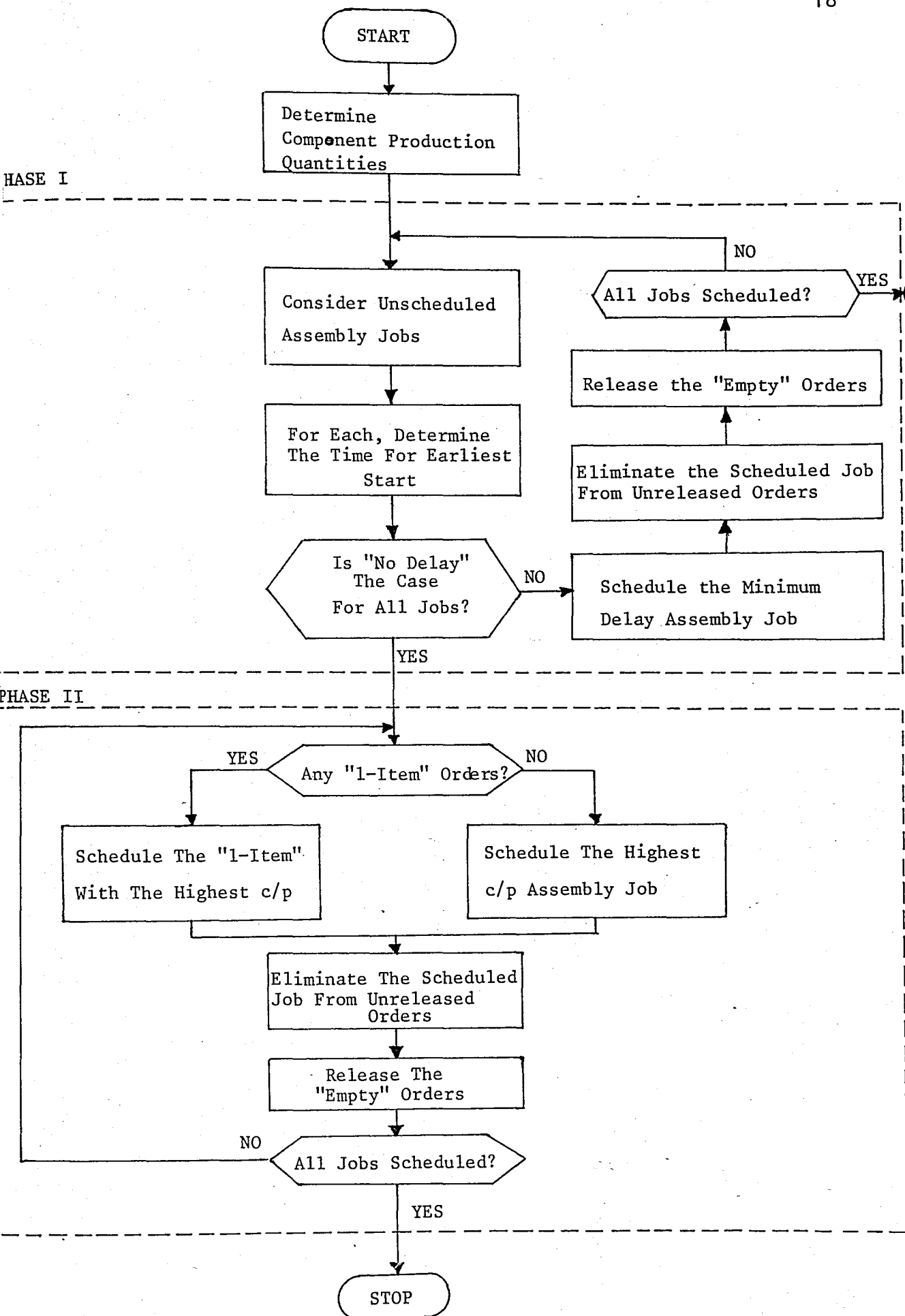


Figure 2.3.2 Flowchart for the 3-level Scheduling Algorithm

also scheduled in the assembly shop. If a tie occurs at this step, it is broken in favor of the candidate schedule with the greatest c/p assembly job.

This procedure aims at precluding idle time intervals at the assembly level. It serves its cause through step-wise minimization of inter-assembly durations. The final objective of minimizing mean flow time of capital is indirectly attended since any idle delay at the assembly level retards all waiting assembly jobs, and delayed assembly of end-products translate release times of orders. Another option for mold/machine and assembly scheduling could be to respect c/p criterion alone. But computational experience reveals that uncontrolled idle times at the assembly shop may bring about much more loss than the advantage gained by virtue of FCR algorithm (see Section 2.4; results with algorithm A2).

Before proceeding further with PHASE ONE, let us investigate the mold/machine scheduling algorithm.

Through seemingly a minor part of the overall algorithm, the problem of scheduling molds on molding machines is quite critical in view of the recent discussion about the consequences of delays at assembly level. This problem is defined as follows.

x molding jobs are to be done on m machines. Each mold j is distinct, and it can only be mounted on k_j of non-identical machines. There are no precedence relations among molding jobs. Ready times of machines are in general different. The jobs are available at any time. The aim is to minimize the maximum job completion time among machines, so that the

associated assembly job can start as early as possible (see Assumption iii above).

A similar problem known as "multi-processor scheduling problem" has been subject to many articles in the literature. It is defined in Nichols et al. [5] as, "... scheduling n independent, single operation jobs, all available at time zero, on m identical processors ... each job must be processed by exactly one of these processors, and it is desired to minimize makespan."

The assumption of "identical processors" is the main deviation between the two models. Besides, the problem dealt in literature implicitly assumes that all processors are ready at the same time.

Still, the algorithm developed in our study is inspired from the "longest processing time" algorithm proposed by Greenberg [6] for that problem.

Figure 2.3.3 gives the flowchart for the algorithm developed to tackle the current problem. Ties that may occur either during job selection or during machine selection can be broken arbitrarily.

The first stage of the 3-level scheduling algorithm, thus concentrates on the lowest and middle levels, namely mold/machine and assembly schedules. However, decisions are made with due consideration about their consequences to the overall objective. It can be said that PHASE ONE presents a hierarchical framework in which two lower levels interact during fixing their respective scheduling decisions while the highest level decisions are let to float.

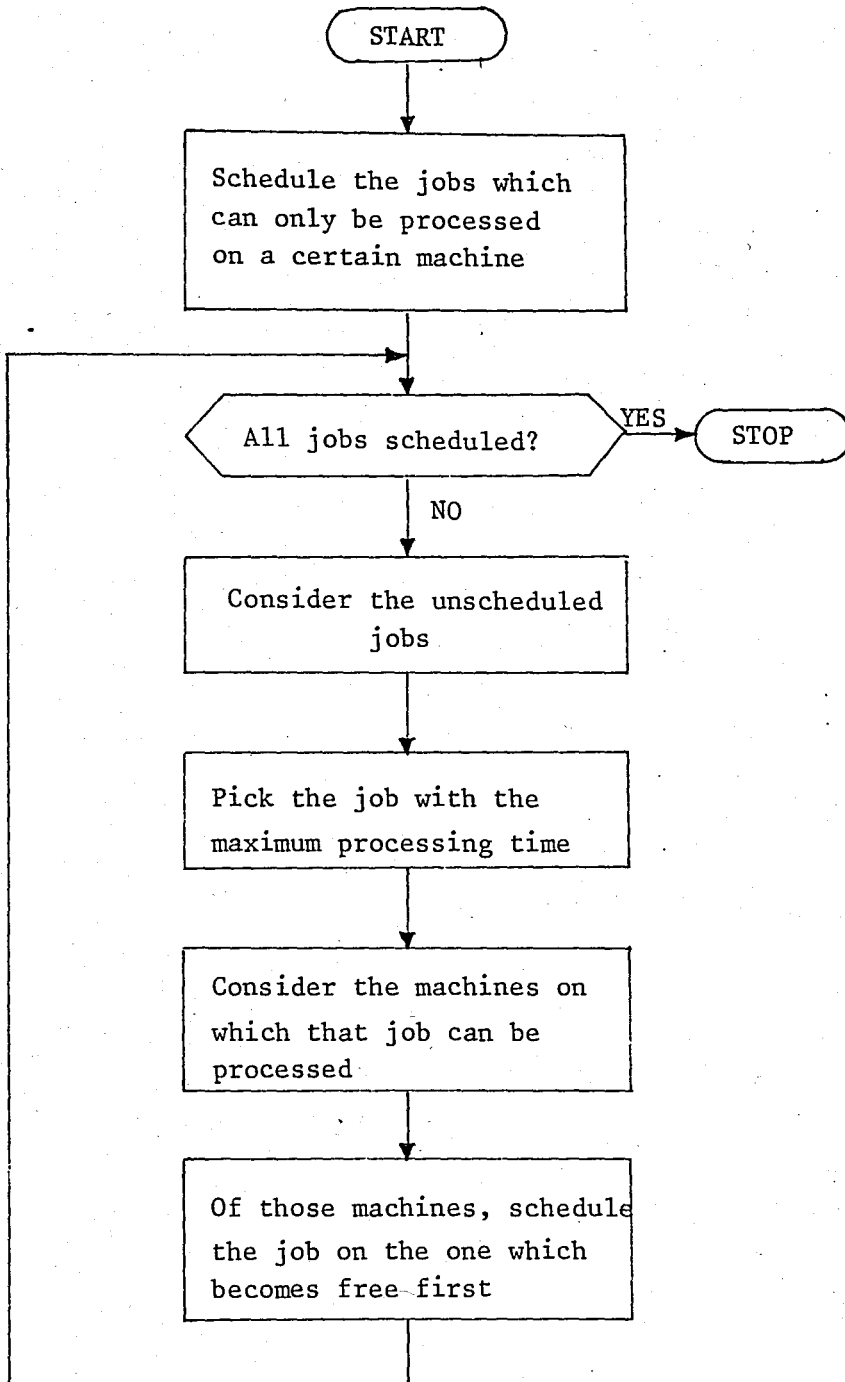


Figure 2.3.3 Flowchart for the Mold/Machine Scheduling Algorithm for a Particular Assembly Job

PHASE ONE closes as soon as there exists sufficient evidence that the assembly shop will never be idle then on, no matter what the order among the remaining assembly jobs will be. That evidence is having all candidate mold/machine schedules letting their respective assembly jobs start simultaneously as the former job ends.

2.3.3.3 Phase Two

If there are assembly jobs which remain unscheduled as PHASE ONE ends, the algorithm enters a new stage where a new hierarchical framework is valid.

In PHASE TWO the focus of attention is on the two upper levels, namely the assembly schedule and the order release schedule.

The main concern is releasing capital as early as possible. Thus, it is an appropriate policy to assign priority to assembly jobs which, when completed, let completion of partially prepared orders. If there are no orders waiting for a single assembly job, FCR algorithm is applied to select among unscheduled assembly jobs the one to be scheduled next. If there exists more than one such order, ties between the awaited assembly jobs are again broken by the FCR rule. When two or more "single item" orders wait for the same last scheduled assembly job, their releases need to be sequenced. They are released with respect to

(total value)

(assembly time required by the demand for the assembly job)

ratio. The order with the higher ratio is released earlier. It is clear that the FCR principle is directly applied in this sequencing decision.

The algorithm terminates as soon as all assembly jobs are scheduled, which coincides with the moment all orders are released.

Similar to the case in PHASE ONE, the second phase of the algorithm involves decision interactions between two levels of the problem. But there is a shift of focus between the two stages. This feature of the algorithm qualifies it as incorporating a "dynamic hierarchical" structure.

As a matter of fact, the assembly schedule is always under consideration. In both phases, its decisions are determined in close contact, first with the mold/machine schedule, and later with the order release schedule.

The decision rule for the isolated assembly schedule could be provided by the constructive algorithm FCR. But constructing the remaining two schedules under dominion of the FCR sequence at the assembly level would, in general, produce poor results in terms of the overall objective of minimizing mean flow time of capital (See Section 2.4; results with algorithm A2). However, FCR rule is put into action at a number of steps, especially with greater weight towards the end of the algorithm.

An alternative to the suggested hierarchical structure could be to shift the attention from assembly level to order level and to fix all scheduling decisions through step-wise minimization of inter-release times of orders. In general,

this algorithm will not perform better due to loose control over inter-assembly-job delays and total negligence of c/p values during assembly scheduling. It will fail more as the molding rate goes ahead of assembly rate, and as the orders are more selective.

2.4 Tests on the Algorithm

The 3-level scheduling algorithm described above can be questioned from two main aspects.

The first question may arise for the design decisions. Couldn't there be a better performing hierarchical structure? Is the mold/machine scheduling decision rule the best fit for the case?

It is also questionable whether the established structure of the algorithm is justifiable in terms of its performance under various circumstances. With what sort of data the algorithm perform more successfully?

Below is the test performed to provide hints to these questions.

2.4.1 Factors

Five factors are investigated in this study;

A) Four Alternative Algorithms;

A1 : The adopted 2-phase algorithm,

A2 : An algorithm in which other two level scheduling decisions are imposed by the FCR (i.e. non-increasing c/p) ordering of assembly jobs at the middle level,

A3 : The adopted 2-phase algorithm without FCR (i.e. c/p) preferences among assembly jobs.

A4 : An algorithm in which other two level scheduling decisions are imposed by the arbitrary ordering of assembly jobs at the middle level.

B) Three Mold/Machine Scheduling Rules:

B1 : The adopted, "longest processing time first" rule,

B2 : "Shortest processing time first" rule,

B3 : Random rule.

C) Three sets of Orders:

C1 : Four orders all of which demand from all items,

C2 : Four orders two of which demand from all items,

C3 : Four orders none of which demand from all items.

(Item-wise sums of demands of all three sets are equivalent.)

D) Two sets of Assembly Processing Times:

D1 : Short,

D2 : Long ($3/2 \times$ Short).

E) Two sets of Ready Times for the Assembly Shop and for the Molding Machines:

E1 : All the same,

E2 : All different.

2.4.2 Tests and Results

Following are the descriptions and the results of the performed tests. The result figures are the mean flow times of capital for the associated runs.

1) B = B1 , D = D1 , E = E1

	C1	C2	C3
A1	435.4	407.9	393.8
A2	478.5	472.0	466.5
A3	435.4	421.2	413.0
A4	471.3	471.1	470.1

2) B = B1 , D = D2 , E = E1

	C1	C2	C3
A1	584.4	543.4	522.4
A2	643.5	633.7	625.6
A3	584.4	563.2	551.0
A4	636.3	636.0	634.6

3) B = B3 , D = D1 , E = E1

	C1	C2	C3
A1	453.4	425.9	411.8
A2	482.7	476.2	470.7
A3	453.4	408.8	384.1
A4	482.7	482.5	481.5

4) B = B3 , D = D2 , E = E1

	C1	C2	C3
A1	602.4	561.4	540.4
A2	647.7	637.9	629.8
A3	602.4	535.8	498.9
A4	647.7	647.4	646.0

5) A = A1 , C = C2 , D = D1

	E1	E2
B1	407.9	441.5
B2	424.8	458.2
B3	425.9	462.4

6) A = A1 , C = C2 , D = D2

	E1	E2
B1	543.4	579.2
B2	560.3	593.7
B3	561.4	597.9

2.4.3 Observations

Upon investigating the above tabulations, the following observations can be made.

1. With respect to tables 1,2,3 and 4, the adopted 3-level scheduling algorithm A1 performed most successfully when run with the adopted mold/machine scheduling rule, A3 performed better than A1 when run with the random rule.

2. With the adopted mold/machine scheduling algorithm, employing FCR criterion has a more pronounced effect when the algorithm is applied for a non-homogeneous set of orders (Tables 1 and 2, comparison of A1 and A3 entries), and/or with longer assembly processing times (Comparison of Tables 1 and 2 for the respective differences between A1 and A3 entries).

3. With the adopted 3-level scheduling algorithm, the "longest processing time first" rule performed most successfully among the three alternative mold/machine scheduling rules when run with two different sets of ready times and with two different sets of assembly processing times (Tables 5 and 6). Upon comparison of tables 1 and 3, and, 2 and 4, in general it performed better than the random rule with the exception of entries A3-C2 and A3-C3 in tables 3 and 4.

4. No significant variation of performance superiority is observed among mold/machine scheduling algorithms with respect to variations in ready times and length of assembly processing times, when run within the adopted 3-level scheduling algorithm (tables 5 and 6).

2.5 Implementation of the Algorithm

The 3-level scheduling described in the previous section is coded in FORTRAN IV. The computer program is prepared in two versions; one to be run in CDC system at Boğaziçi University, and one to be used in IBM PC/XT micro-computer of the company.¹

¹ Data files and output listings of the programs mentioned in this thesis can be referred at the IE Department archives.

Besides schedules determined by the algorithm, the program is prepared also to provide supplementary information on

- (A) the requirements for metal and gasket components,
- (B) the color distribution of plastic components and armature production,
- (C) the initial and final levels of the stocks, the required and produced quantities of the plastic components.

The firm uses the program to facilitate periodic production scheduling. It is also considered as a tool to investigate results of various scenarios such as grouping orders and scheduling once for each group.

2.6 Possible Extentions

This study can be interpreted as a rudimentary treatment of an interesting production scheduling problem. Though it involves three distinct stages of a process, its conduct is not a comprehensive one in view of unmentioned functions peripheral to production.

In a broad outlook, there are three main organizational units which interact closely with production. They are; material requirements planning, inventory planning and distribution planning.

The approach in this study has been to focus on production, and develop a modular model that can fit and perform successfully in the overall production planning activity.

As an extension to what has been done, the principles proposed for multi-level scheduling can be applied to a similar system in a manner incorporating the interactions mentioned above. Such an implementation could involve a system-dynamical framework, introducing feedback loops, and controls on operational variables. For instance, results of scheduling decisions may suggest modifications on lot sizes or on parameters of the inventory policy. Similarly, resulting requirements for externally supplied components could affect material requirements planning decisions more intimately. More effective distribution of products could be achieved through considering the regional origins of orders at the order release scheduling level.

III. AN IMPLICIT ENUMERATION ALGORITHM FOR THE ORDER SELECTION PROBLEM

3.1 The Problem

This part of the thesis deals with the very common problem of selecting among all orders those to be answered simultaneously given that on-hand inventory is short of meeting all of them.

This problem too, was faced by the firm discussed in chapter one. It will be recalled that orders were being received from wholesalers on a periodic basis, and they were made up of demands for up to nine types of products. With the current policy, the inventory is sufficient to satisfy some of these orders momentarily. The remaining, together with the stock-replenishment order, are met as production proceeds.

The problem is relevant to the concept of capital flow rate analyzed in chapter one. The selection should be made

in a manner minimizing mean sojourn time of capital bound to finished products. Thus the total capital contained in the combination of orders selected for simultaneous release should be maximum. Of course, subject to the constraints imposed by available stocks, and that no order can be satisfied partially. It is apparent that what we are up to is a problem of combinatorial nature.

Let us formulate;

$$\begin{aligned} \max \quad & \sum_{i=1}^m \sum_{j=1}^n c_j o_{ij} x_i \\ \text{subject to} \quad & \sum_{i=1}^m o_{ij} x_i \leq I_j \quad j = 1, \dots, n \\ & x_i = 1 \text{ or } 0 \quad i = 1, \dots, m \end{aligned}$$

where c_j : capital contained in product j ,
 o_{ij} : demand of order i for product j ,
 x_i : 1 if order i is selected, 0 otherwise,
 I_j : available stock of product j .

This formulation reveals that our problem is a typical 0/1 multidimensional knapsack problem.

Literature on this special problem of integer programming contains articles proposing heuristic methods to obtain approximate solutions (e.g. Loulou et al. [7]).

Our approach will be to develop an efficient implicit enumeration algorithm exploiting the nature of the problem to find the exact optimal solution. This approach is justified in view of the moderately sized problems which will be attacked.

3.2 An Implicit Enumeration Algorithm

As defined by Garfinkel et al. [8], "Implicit enumeration is the name of a class of branch and bound algorithms designed specifically for the case in which x is required to be a binary vector."

In our case, the decision variable vector x is binary, in that its elements can either be 0 or 1.

Before proceeding to describe the algorithm devised for the order selection problem, let us define the nomenclature.

- c_i : capital content of order i ,
- D : set of free variables dropping to 0 due to reduced inventory,
- F_k : set of free variables at vertex k ,
- FATH(k) : TRUE if vertex k is fathomed, FALSE otherwise,
- I_k : remaining inventory vector at vertex k ,
- I_k^j : inventory for item j remaining at vertex k ,
- LB : global lower bound on the objective function value,
- LOW(k) : lower bound on the tree emanating from vertex k ,
- m : number of considered orders,
- N : maximum number of vertices,
- NV : number of created vertices,
- O_i : demand vector of order i ,
- o_{ij} : demand of order i for item j ,
- OPTSOL : set of selected orders for the optimal solution,
- OPTV : optimal vertex,
- p : the partitioning variable,

PARV(k) : the partitioning variable chosen at vertex k,
 PV(k) : the vertex parent to vertex k,
 RMV : the rightmost vertex,
 S_k^0 : set of variables of which values are 0 at
 vertex k,
 S_k^1 : set of variables of which values are 1 at
 vertex k,
 UP(k) : upper bound on the tree emanating from vertex k,
 V : the number associated with the current vertex,
 VV : the number associated with the previous vertex
 while branching right.

Figure 3.2.1 gives the complete description of the algorithm. However it will be useful to recount shortly.

The procedure starts from vertex 1 where no variable has an assumed value. It progresses by partitioning the set of possible solutions through ramifications to right and left; the right branch assuming the value 0 and left branch 1, for the chosen "partitioning variable." Branching advances first on the leftmost side until a vertex is reached with $F = \{ \}$; in other words until it is fathomed. Backtracking searches for the solutions suggested by the unattended right branches. A right vertex is fathomed if the upper bound on it is less than or equal to the global lower bound developed so far. Any vertex of which both branches are fathomed, is itself fathomed, too. The algorithm terminates when there remains no unfathomed branches, that is when the rightmost branch is fathomed. The order selection problem has a feature which helps prune many branches and thus complete the enumeration with small number

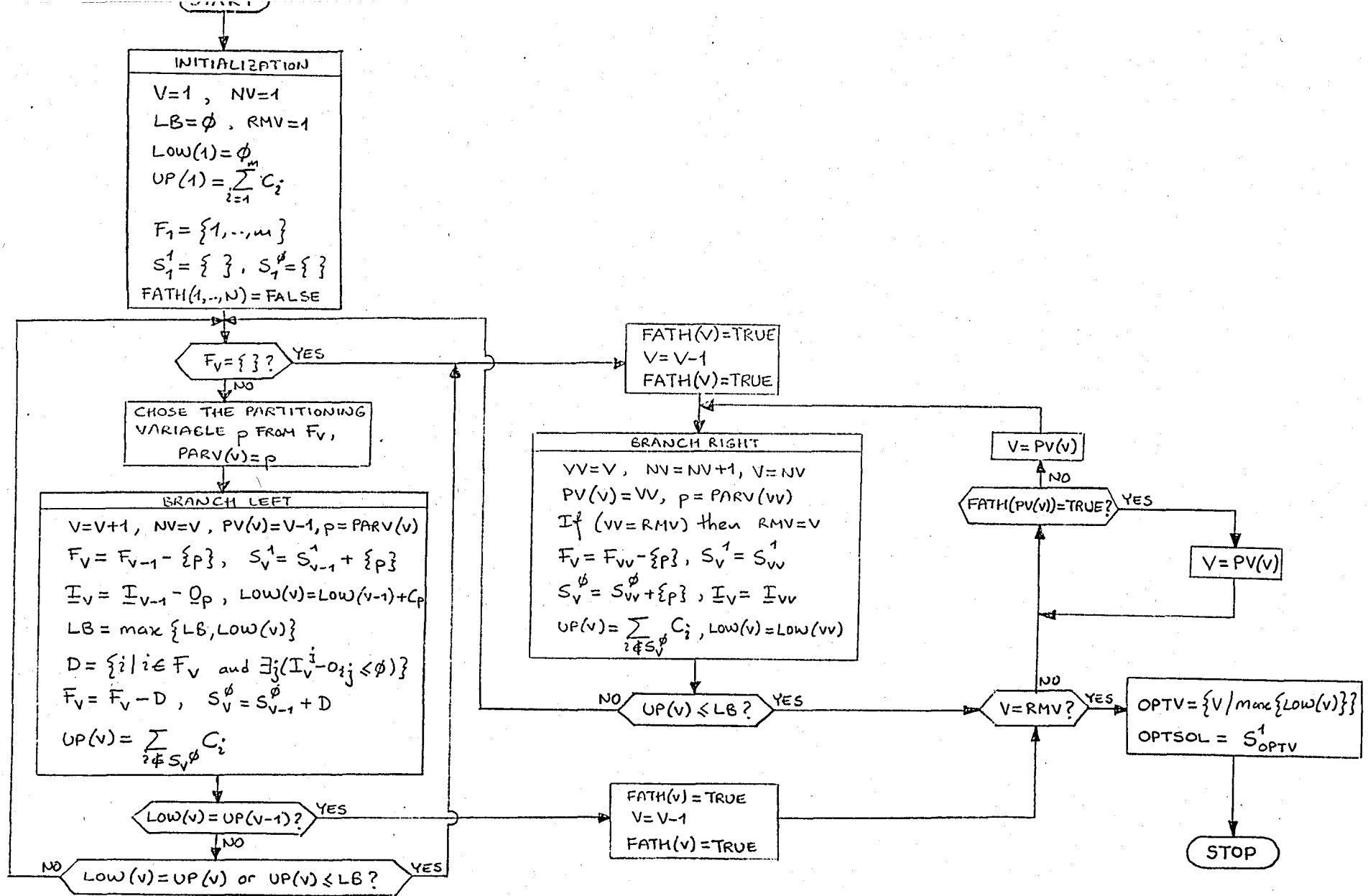


Figure 3.2.1 Flowchart for the Implicit Enumeration Algorithm

of vertices. It is possible to assign the value 0 to certain variables by comparing their associated item demands with the remaining inventory at the current vertex. If the available inventory is short of satisfying the demand for at least one item, the variable, lifting need to branch further, assumes the value 0. In figure 3.2.1, D is the set of such variables.

The performance of the algorithm depends heavily on the method used for the selection of the partitioning variable among free variables. A proper selection at an early stage may significantly reduce the number of vertices needed for algorithm termination.

In our study, eleven decision criteria are suggested for partitioning variable selection. The following sections are devoted to their descriptions and comparison of their performances under a number of circumstances.

The algorithm is coded in FORTRAN and run in CDC system in Boğaziçi University.

3.3 The Proposed Heuristics for the Selection of the Partitioning Variable

Silver et al. [4], while classifying heuristic methods, note that, "... a heuristic may be used as a part of an iterative procedure that guarantees the finding of an optimal solution." One possibility for such use of heuristics may be, they continue, "... to make a decision at an intermediate step of an exact solution procedure, e.g., the rule for selecting the variable to be entered into the basis in the Simplex Method is heuristic in that it does not

necessarily minimize the number of steps needed to reach the optimal solution."

Decision rules for the selection of the partitioning variable at a vertex in the algorithm described above are also heuristics of this kind. No matter for which variable we branch at a particular vertex, sooner or later we reach the optimal solution. However, it is important to keep the number of vertices small in practice, since the memory required by the prepared computer program varies directly with this quantity.

Following are eleven such heuristics.

- 1) LARGEST TOTAL VALUE FIRST (LTVF)
Choose the free variable with the maximum capital content.
- 2) SMALLEST TOTAL VALUE FIRST (STVF)
Choose the free variable with the minimum capital content.
- 3) MEDIAN TOTAL VALUE FIRST (MTVF)
Choose the free variable with the median capital content.
- 4) DEMANDING THE LEAST REMAINING ITEM LEAST FIRST (DLRILF)
Choose the free variable which demands the item remaining least in the inventory, least.
- 5) DEMANDING THE MOST REMAINING ITEM LEAST FIRST (DMRILF)
Choose the free variable which demands the item remaining most in the inventory, least.

- 6) DEMANDING THE MOST REMAINING ITEM MOST FIRST (DMRIMF)
Choose the free variable which demands the item remaining least in the inventory, most.
- 7) DEMANDING THE LEAST REMAINING ITEM MOST FIRST (DLRIMF)
Choose the free variable which demands the item remaining least in the inventory, most.
- 8) DEMANDING THE LEAST ABUNDANT ITEM LEAST FIRST (DLAILF)
Choose the free variable which demands the item with least abundance (i.e. $\text{abundance}(j) = I_v^j - \sum_{i \in F_v} o_{ij}$), least.
- 9) DEMANDING THE MOST ABUNDANT ITEM LEAST FIRST (DMAILF)
Choose the free variable which demands the item with least abundance, most.
- 10) DEMANDING THE MOST ABUNDANT ITEM MOST FIRST (DMAIMF)
Choose the free variable which demands the item with most abundance, most.
- 11) DEMANDING THE LEAST ABUNDANT ITEM MOST FIRST (DLAIMF)
Choose the free variable which demands the item with least abundance, most.

3.4 Results

The eleven heuristic rules are coded in FORTRAN and integrated with the program of the algorithm. The program is run for the following values of problem parameters once for each heuristic.

- A) Inventory Content = I_0
 Number of Products (NP) = 12
 Number of Orders (NO) = 6, 9, 12, 15, 18, 21
- B) Inventory Content = I_0
 Number of Orders (NO) = 12
 Number of Products (NP) = 3, 6, 9, 12, 15
- C) Number of Orders (NO) = 12
 Number of Products (NP) = 12
 Inventory Content = $(1/2)I_0, I_0, 2I_0$

The number of vertices required for algorithm termination are plotted for cases (A), (B), and (C) in Figures 3.4.1, 3.4.2, 3.4.3 respectively.

The following conclusions are drawn;

- 1) Greedy heuristics LTVF, DMRIMF, DLRIMF, DMAIMF and DLAIMF perform significantly better than the rest in all three cases.
- 2) The heuristic LTVF, though requiring minimum of computation with respect to other greedy algorithms, performs best in every circumstance.
- 3) With few exceptions, the algorithm requires increasing number of vertices for completion when run with
 - a. increasing number of orders,
 - b. increasing inventory content, and
 - c. decreasing number of products.

NP = 12

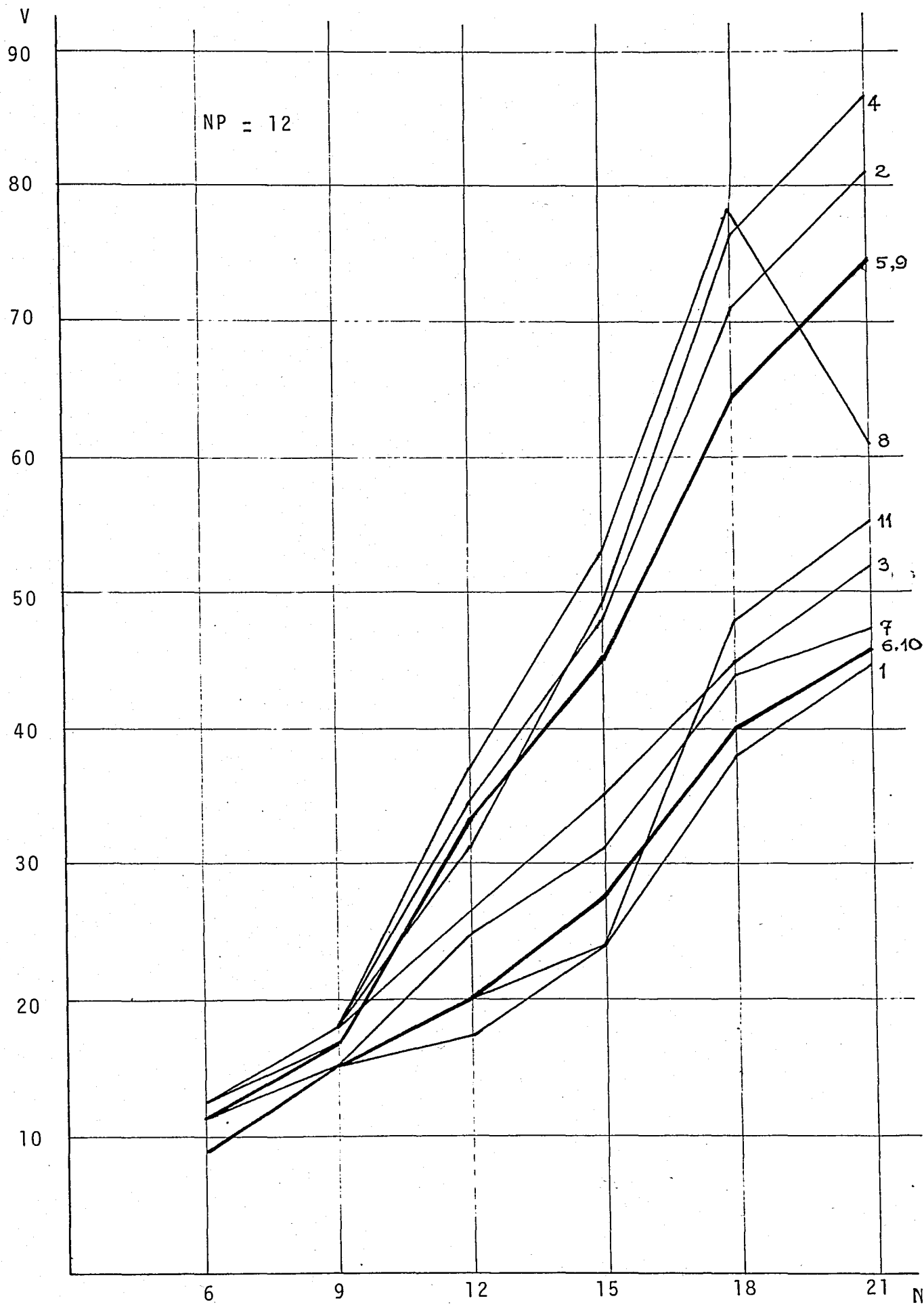


Figure 3.4.1 Results for Case A

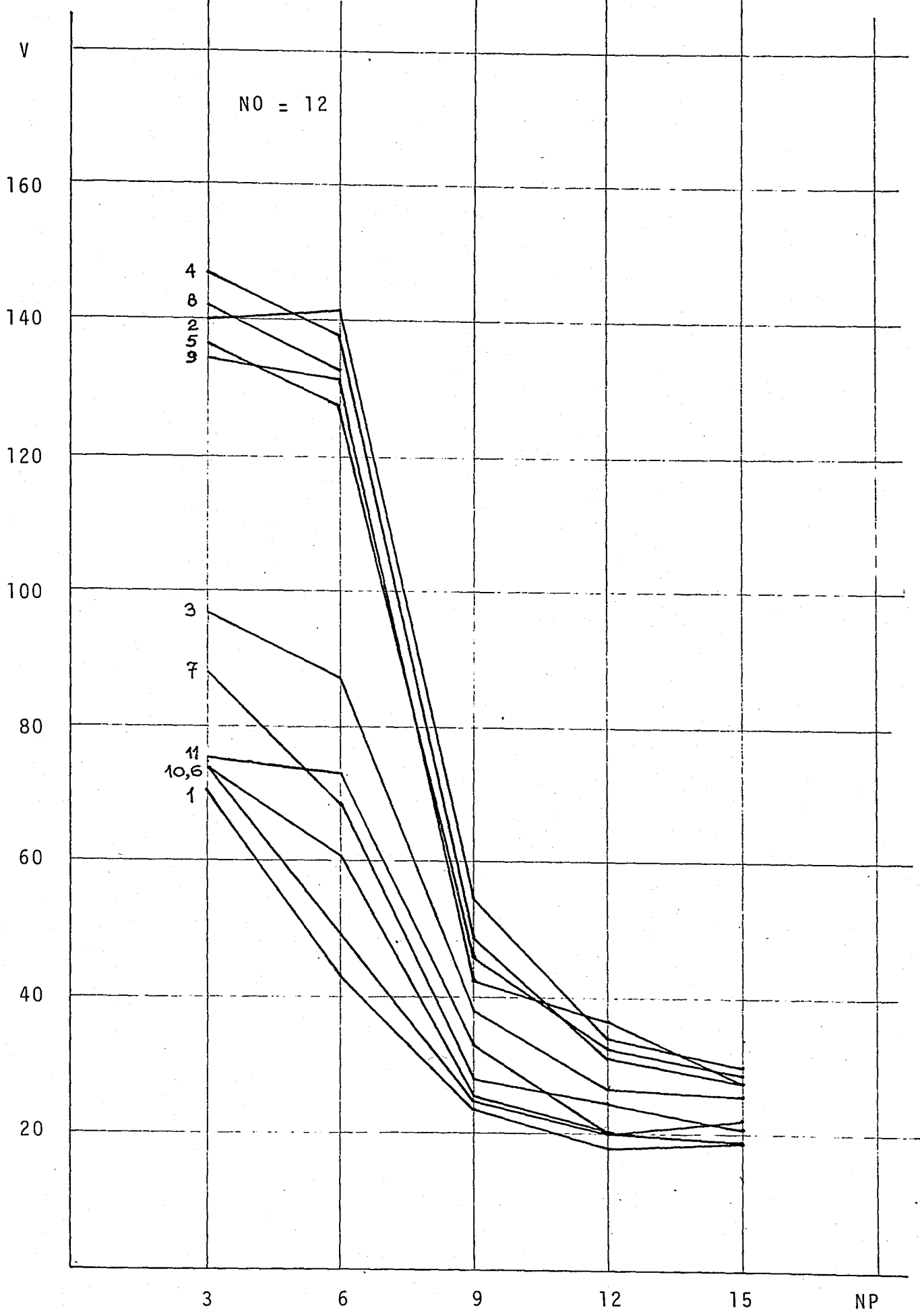


Figure 3.4.2 Results for Case B

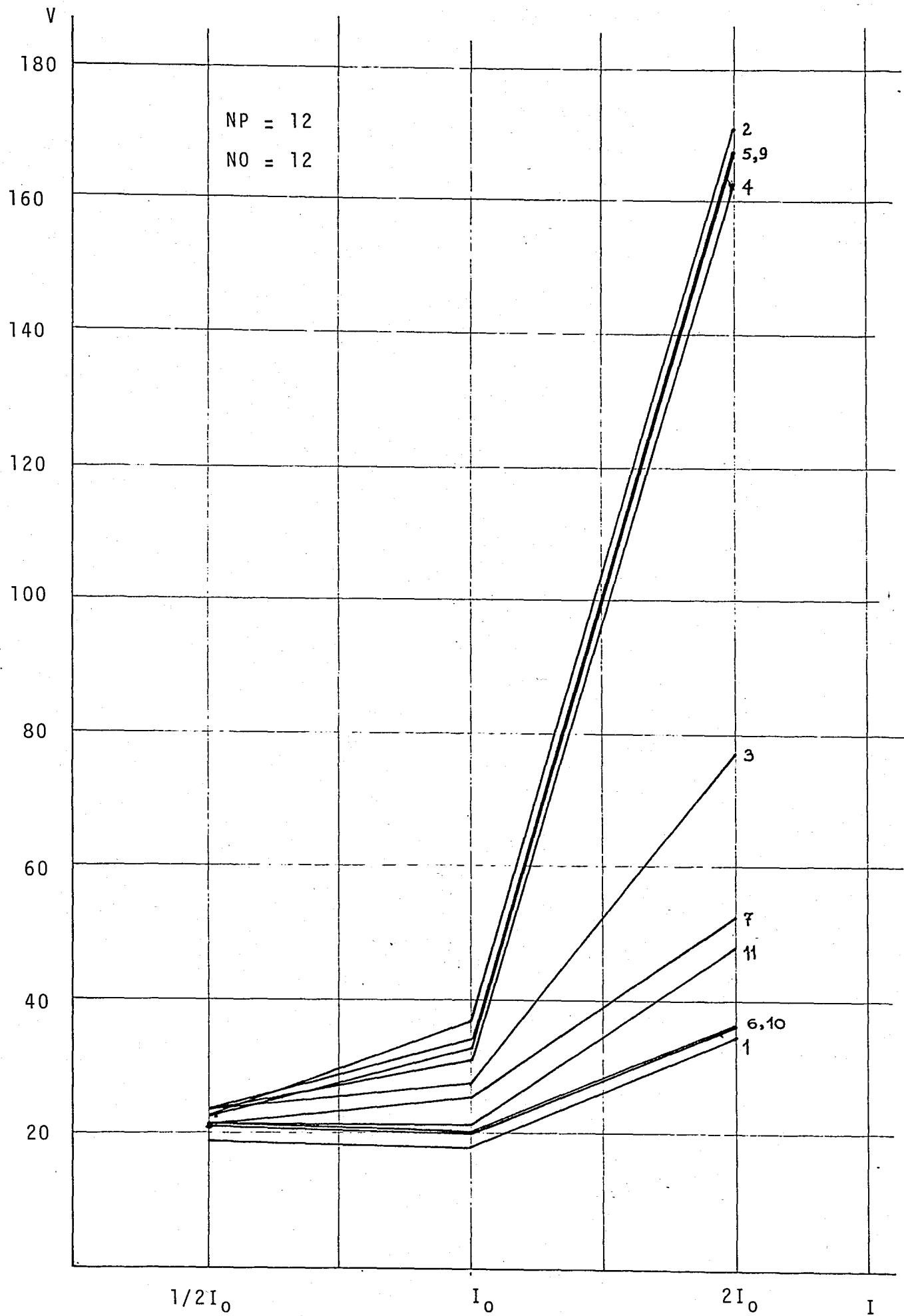


Figure 3.4.3 Results for Case C

These trends make sense since number of feasible combinations of orders is directly proportional to number of orders and available stocks, and inversely proportional to number of products. This point also emphasizes the effectiveness of free variable elimination procedure inherent to the algorithm.

IV. CONCLUSION

In the preceding chapters two distinct problems were analyzed and algorithms for their satisfactory solutions were introduced. The two procedures emerged as two tools that can be used together in the real-life problem treated by this study. Before proceeding with the application, it is worthwhile to note certain points with regard to algorithms themselves.

The 3-level scheduling algorithm has as essence the notion of "fastest capital release". Its elements work in the direction which renders minimum delay of value-containing output. Whether it be the case that stocks of finished products are kept or not, the philosophy of the algorithm is justified since anyhow time is money. If all orders are met by "just-in-time" production, the merit of the algorithm is obvious. In the other extreme, if all orders are met simultaneously from the inventory (which is the classical approach in inventory/production management), the algorithm

hastens the conversion of invested capital into sellable goods.

The implicit enumeration algorithm for the order selection problem likewise minimizes the delay of sales of finished goods through maximizing the total value of simultaneously-released orders. Its very nature presumes existence of end-product stocks. It is applicable to all multi-item whole-selling activities.

One possible application for both algorithms was the case of the producer company we have discussed.

The company had been keeping stocks of finished armatures, yet it was desired to organize production such that some orders could be answered directly as armatures are produced, and thus stock levels could be lowered somewhat.

Thus, as periodic orders are received, those orders selected by the second algorithm can be answered instantly, and the rest (perhaps together with the stock-replenishment order), as the scheduling algorithm decides.

It is clear that if the production capacity is sufficient, the on-hand inventory levels may be kept low. Thus the production manager may adjust the stock levels by considering the limitations of the established system, the trends in demand, and the stock-holding costs. The two algorithms can thus be employed under various stock-holding policies by arranging the frequency and sizes of the stock-replenishment orders.

One can think of many applications for the 3-level scheduling algorithm. Up to this point, it was suggested for periodic production scheduling. It can also be implemented

for dynamic decision making through updating the states of the molding machines and the assembly shop. Thus, the schedules can be revised in view of realized processing times and/or newly arrived orders. Another application may be using it as an aid in making overtime decisions. It is advantageous to arrange overtime working for the molding process so as to eliminate time gaps between consequent assembly jobs. Likewise, overtime decisions for the assembly shop which would let early release of orders can be made more effectively with the aid of the algorithm. The algorithm may also be used to reconsider molding lot sizes in view of their effects on the overall performance of the schedule through simulation.

This thesis introduces algorithms to help solve two distinct problems which may be interrelated in some real cases, as the one treated above. They are useful tools in the sense that they may be used under various operating conditions and in many applications.

REFERENCES

1. French, S. Sequencing and Scheduling. Ellis Horwood Ltd., U.K., 1982.
2. Conway, R.W., Maxwell W.L. and Miller, L.W. Theory of Scheduling. Addison-Wesley, USA, 1967.
3. Müller-Merbach, M., Heuristics and their design: a survey, *EJOR*, 8, 1, 1981.
4. Silver, E.A., Vidal, R.V.V. and de Warra, D., A tutorial on heuristic methods, *EJOR*, 5, 153, 1980.
5. Nichols, R.A., Bulfin, R.L. and Parker R.G., An interactive procedure for minimizing makespan on parallel processors, *IJPR*, 16, 77, 1978.
6. Greenberg, I., Application of the loading algorithm to balance workloads, *AIIE Transactions*, 4, 337, 1972.
7. Loulou, R. and Michaelides, E., New greedy-like heuristics for the multidimensional 0-1 knapsack problem, *Op. Res.*, 27, 1101, 1979.
8. Garfinkel, R.S. and Nemhauser, G.L. Integer Programming. John Wiley, USA, 1972.

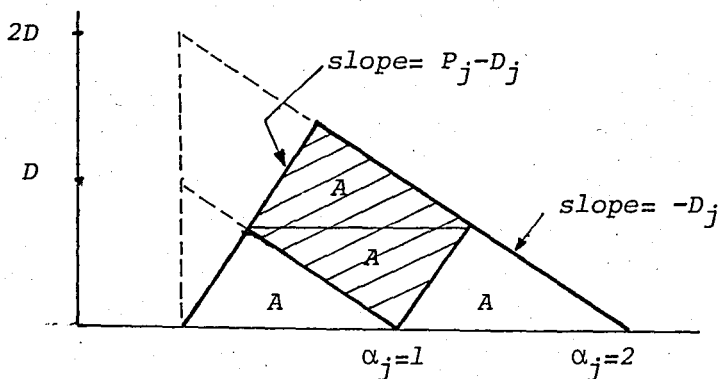
APPENDIX

DETERMINATION OF ECONOMIC LOT SIZE FOR MOLDING

Nomenclature :

- π_j : probability of damage while mounting mold j
 C_j : cost of damage for mold j (TL)
 D_j : average periodic demand for component j (units/period)
 P_j : production rate for component j (units/period)
 h_j : cost of holding component j (TL/unit-period)
 α_j : number of periods of which demands make up the lot size for mold j.

Derivation :



$$A = \frac{1}{2} D_j \left(1 - \frac{D_j}{P_j}\right)$$

$$\text{Average on-hand inventory} = \alpha_j A = \frac{1}{2} \alpha_j D_j \left(1 - \frac{D_j}{P_j}\right)$$

$$\text{Holding cost for a period} = \frac{1}{2} \alpha_j D_j \left(1 - \frac{D_j}{P_j}\right) h_j$$

$$\text{Expected damage cost for a period} = \frac{1}{\alpha_j} \pi_j C_j$$

$$\text{Total cost for a period} = \frac{1}{\alpha_j} \pi_j C_j + \frac{1}{2} \alpha_j D_j (1 - D_j/P_j) h_j$$

Necessary condition for minimum total cost;

$$\frac{\partial TC_j}{\partial \alpha_j^*} = - \frac{\pi_j C_j}{\alpha_j^{*2}} + \frac{1}{2} D_j (1 - D_j/P_j) h_j = 0 ,$$

$$\alpha_j^* = \sqrt{\frac{2(\pi_j C_j)}{D_j (1 - D_j/P_j) h_j}}$$

Sufficiency condition;

$$\frac{\partial^2 TC_j}{\partial \alpha_j^{*2}} = \frac{2\pi_j C_j}{\alpha_j^{*3}} > 0$$

Hence α_j^* gives the lot size yielding the minimum total periodic cost.

α_j^* should be truncated or raised to an integer for practical reasons. This decision can be made by comparing the total costs caused by the two alternatives.