# SIMULATING SCHEDULING PLANS

C. E. Montagnon
The Dunlop Company Limited
Engineering Group
Coventry, England

Abstract

This model simulates the activities of planning and manufacturing batches of components in a job shop. Simulation of the planning stage forms the basis of a tool to completely reschedule the shop from its actual position, having regard to due dates and machine resources. The execution of this plan is simulated giving management data on the effects of present decisions. A significant feature of the model is the use of simulation of present clerical rules as a basis for a more powerful shoploading system.

## 1. INTRODUCTION

How to plan and control the passage of components through a job shop is the problem for which this paper reports one solution. The area concerned is the taking of the raw material, machining batches of components, and passing the components into a stores. Assembly problems are omitted from the model and the decision of how large each batch should be is assumed made previous to the steps covered by this model.

In loading a machine shop, two main problems arise:

(1) How shall the operations of the new batches be planned so as to try to meet the due date within the restrictions of the work already committed?

(2) How can the shop be reorganised to account for the work getting away from the original plan?

If, as was the aim here, a completely computerised production control system is planned, then programs to solve these two problems must be written.

Usually in specifying computer programs the processing is comparatively straightforward and the main attention has to be paid to the formats of input output, but in this case one does not even know what the processing routines should be! So the main aim of this work was to find routines that would answer these loading problems.

The second aim was to make this a starting point for a large model of all the main activities of the Company so that the effect of any management decision on the whole structure could be simulated. Thus a first extension of a model such as this might be to include the assembly problems at one end and the batching problems at the other, referred to above.

First attempts are rarely the best, so a modular approach is taken to building this model. Thus each part of the model is to be made to conform with a standard interface with those parts around it. Then if an improvement is found for any part, it can be replaced at a later date without disturbing the whole model. Another reason for developing standard interfaces like this is that the user department gets used to one sort of output, and if

the procedures are improved later, the quality of their output improves but the format does not change. Thus if they can be convinced to accept the initial simple system because they can understand it, it will not be necessary to explain to them how a more sophisticated system works when it is introduced.

Simulation has been used here as a first attempt to solve the problem for two reasons:

(1) Programming the logic of a process will require the members of the computer department's team to thoroughly understand what is happening.

(2) If the present clerical decision procedures can be simulated then the principle that a computer can work much faster than humans can be used to find totally new applications of these procedures.

Scheduling plans should not be judged from the value of the objective that they plan to achieve, but the value of this objective that is actually achieved when the plan is enacted. This belief led to the construction of a model of the machine shop itself to test the effectiveness of the various plans produced by different schedulers.

The size of the problem is about 3000 batches on the factory at any one time, with an average of 10 operations each, and about 100 multiple machine centres. Thus a first look at the problem suggested it was too large for a direct analytical attack and that simulation was the best approach for a first attempt.

## 2. OVERALL MODEL STRUCTURE

The main flow of data is outlined in figure 1, and more detail is shown in figure 2. Thus broadly speaking, the model takes in a report of the current status of the shop, reschedules to produce a plan, and simulates that plan to estimate what will happen.

In figure 2, the information on the left must come from the user, with that which is needed anew every run shown boxed in, while that which is more permanent shown open. The system provides the reports shown on the right-hand side. Each of the six processes have been made modular to enable easy replacement. In the application reported here the setting up of the component library took nearly nine months, and since it was obviously the critical activity, it was begun first, before anyone had any idea how the final system would look.

An attempt has been made to separate those procedures that are based on local factors, e.g., the pattern of the component library, the form of data collection on completed operations. The scheduler and machine shop simulation (boxes 4 and 6) have been written in general terminology as they are by far the hardest parts to write and hopefully can be used as they stand in another similar situation.

The system outlined above does not cope with the new batches but they are dealt with in a similar way. First their details are translated to the general environment (as in box 2), then they undergo a schedule to produce reports similar and in addition to the ones for the complete reschedule.

## 3. SCHEDULER

To understand this, it is easiest to follow the way it was built. This was:

(1) Observe the clerk loading the new batches.
(2) Note down all his decisions.
(3) Recognise a pattern in these.
(4) Formulate the "rules" he is using, in his terminology.
(5) Order the rules for a computer program.
(6) Program these rules.
(7) Test the rules against the clerk's own data and work.
(8) Abstract and generalise the rules.
(9) Full scale program to test scheduling of new work.
(10) Amend the rules to cater for current work.
(11) Reprogram to cater for current work.

Most of these steps are self-explanatory. Here one has gone from the particular to the general, taking the highly "local" experience of the clerk and using this as a foundation for a much more powerful model. The process of abstracting and generalising the rules gave a much greater insight into what the rules were actually doing; for instance, how they catered for transport times.

The objective of the scheduler is to take all the operations of the batch being considered, form these operations into groups where one group represents the work to be done in one week and to fit these assigned weeks within the capacity still available and within the time to the due date. The rules are selected in turn to see if this can be achieved, with each rule asking for more and more to be done in one week. This is really heightening the priority of the batch as it becomes apparent that there is very little spare time available. When a grouping, or week sequence, for the operations is suggested that is within the time available, the operations are then tested for available capacity and the whole sequence loaded around the crucial operations. This is illustrated in figure 3.

As an example of the rules, take the following one expressed in local terminology

"Add to previous groupings by allowing there to be up to four operations from the same machine centre together. Only add to groups whose time is less than 21 hours and the total new group time is not to exceed 25 hours".

What this rule is saying is that, now this batch has become as critical as it is, one can reduce the time allowed for inspection and transportation between the operations and so get more operations in one week. Expressed in generalised form this rule becomes:

"Allow the same type operations to be in one group as before but now also allow up to $P_1$ operations which share property 1 in one group. Do not add to groups of time more than $T_1$ or extend groups to time more than $T_2$".

It is readily seen that this rule is much more flexible than the local form above and in particular the "property" could be anything, for instance using the same tool.

The data environment has forced the application of rules where transportation and set up are only considered as broadly influencing the parameters of the rules, since details about such values and the tools needed were not originally available. The only data that was available was an estimated standard time for each operation. Set up was catered for by only loading to a certain percentage of the hours available, while transportation was catered for by the sort of rule shown above. Tools are not directly considered except that each batch is given a current earliest start date which may reflect the lack of tooling. In fact, all these levels and values are entered through the parameter values given to the scheduler, so that should such data become available later on it would only really change the local procedures such as in box 2 in figure 2; tools would simply be treated as an extra resource by the scheduler.

As shown above, the "rules" were originally developed to cater only for new batches. But in using these rules on new batches three points became apparent:

(1) In order to schedule new work a <u>correct</u> statement of the committed work is needed. Since the work inevitably falls behind the plan made for it when it was new, it is not sufficient to base the current estimate of committed work on the assumption that operations were performed when planned. Periodically, the whole of the outstanding work should be rescheduled from the situation it is actually in.

(2) When lots of jobs get a long way behind on the shop it becomes confusing to know what to do next. So a new plan is really needed.

(3) In principle, there is little difference

in scheduling the outstanding operations of current work to that of scheduling new work. This task is quite impossible for a clerk to carry out in reasonable time so here is an opportunity to apply the standard computer technique of doing something so much faster than a human that a new application of the same ideas is possible.

Thus one is led to take these very same rules, sort all the outstanding jobs by some critical value (the current one used is the due date) and take the shop as completely empty and use the rules to load all the work onto the shop again. Thus the most critical job will find the shop capacity fully available. This is what the re-scheduler actually does. Naturally, if the scheduler is realistic, as time progresses points (1) and (2) will no longer be so relevant and the period of rescheduling can increase.

## 4. MACHINE SHOP SIMULATION

A schedule produced by such an algorithm as above will not turn out exactly as planned in practice, because of such things as: step-up time is not considered correctly; the times are only estimates; the human element has not been considered; the tools may not be available, etc. Thus any evaluation of a schedule should not be on what it schedules to happen, but on what actually does happen. One way to try to estimate this is by simulation. As a first attempt at a simulation, many similar assumptions to those in the scheduler are made, which tends to make the simulation of little apparent value, but even such a simple simulation does have worthwhile points:

(1) It sets the logical framework for adding in the complications later on.

(2) The substantial programming problems involved in such a large simulation can be sorted out in a comparatively simple environment.

(3) It is possible to test whether it is worth having a plan such as the one above at all.

The main events that this simulation considers are: machine broken; machine repaired; job arrived; job transported between operations; operation completed; job completed; and simulation end. The machine centres have several machines, each of which is individually loaded. (This is not so in the scheduler where an n machine centre is considered to be one machine of n times the power). Set up time is not considered directly, but if it is not included in the required operation times then it can be accounted for by reducing the time available in a week. Transport time is modelled, whereas it is only included indirectly in the scheduler. The operation times used are the standard times used in the scheduler. This is so since no estimates are available at the moment of how the distribution of actual times is related to this. Also it is felt that if the estimated time is the mean, then if one is only going to run the simulation once to achieve values of certain measures, using the mean for all operation times will give a good estimate of the mean of the measure.

With such a large problem (3000 jobs and 100 machine centres) the main difficulty in building a simulation is in fitting the program within the space and time available. When this is achieved with a reasonable real time to run time ratio, it is then possible to enter into a dialogue with the machine shop to improve the model in a similar way as was done in the scheduler.

The basic logic of the model is a series of activities, each representing what happens at one of the events outlined above. The time for the next event on any of the entities, machines, repair centres, and jobs, is recorded and time advanced to the minimum one. Then the cycle of activities is investigated for every entity to see if that activity is about to occur. One point of the logic is worth noting. When a job has been transported from its previous operation it is ready for its next. If the machine for this is available, were it to be started, it might make the machine

unavailable for another job, lower in the job list, which is also about to arrive at the machine, but has a higher priority. In order to overcome this, jobs are only moved to a machine's queue when they need it, and are not actually loaded on it; there is a further activity that examines each machine centre to see if it has any capacity and loads this capacity from the queue using the priority rule. There are several priority rules for selecting the next job to be done on a machine, such as, first-come-first-served, shortest operation, least slack, etc. The one which really tests the plan is that which takes the job with the earliest scheduled time.

The input to the model consists of such things as number of machines at each centre, average breakdown interval at each machine centre, start time for data collection in model, period for full scale report, etc.

When such a model has been built, care must be taken not to generate too many reports from it. In this case special effort is made to keep the reports small and concise. The general report at the model end gives average lateness, average number of jobs on the machine shop, and the average delay experienced, the average queue length and the percentage idle of each machine centre. When a job leaves the shop there can be a report on it giving its average delay per operation. Periodically, there can be reports on the queues at each machine.

Besides testing a suggested plan, the simulation can be used to highlight the machines at which delays occur or which are idle, and it could be rerun to find the effect of altering the capacities of these centres, thus aiding decision about capital investment. This type of question seems not to be asked by management, presumably because they are not used to being able to have the answer.

## 5. PROGRAMMING FEATURES

### 5.1 SCHEDULER

In the early version of this, the rules were still in local terminology and the component library was referenced in the scheduling program. The "local" form of the rules implied asking "character" type questions and the library had to be accessed randomly. This suggested the use of Cobol. But the computations as regards fitting into available machine capacity and finding the difference between dates in weeks, needed Fortran if they were not to be too heavy. The manufacturer (ICL) had little experience in using these languages together, and there was great difficulty in producing outputs from the two compilers compatible for consolidation into one program.

Abstracting the scheduling rules caused the computational and localised parts of the problem to be separated. Thus in the later version of the scheduler a Cobol program (box 2, figure 2) takes the data from a local environment, performs the necessary library accessing and produces the data in a numeric coded form. Then a Fortran program which is generalised in nature takes in parameters for this particular environment and performs the juggling to produce the schedule. This program is much easier to understand and to amend than the earlier version. In fact, the structure of the program, or order in which the rules are taken, can be altered by simply changing one GO TO statement. Also with very little reprogramming new types of rules could be introduced. Since this program is written in Fortran IV it is easily transferable to other equipment.

### 5.2 SIMULATION

In the first version of this the queuing situation modelled seemed to call for a language like CSL (Control and Simulation Language) which deals with ordered sets and has automatic asynchronos time advancing. Working with such a high level language (a CSL program is first translated into Fortran when being compiled) made the program logic much easier to follow and write. A small scale model was written first with all the data in core and then this was extended to have the job details on a disc backing store. This step was necessary since any of the 3000 jobs modelled might have up

to 70 operations. The job details were available from the disc in a random manner. The main problems that arose with this program were:

(1) For each operation on a job the disc file had to be randomly accessed four times. This slowed the program down a great deal and a day's work in the machine shop took about 25 minutes to simulate.

(2) The program expanded at run time as the various queues were filled up and this expansion was so much that it did not seem possible to run a problem of more than 2,500 jobs. For although the program started at 18K words it would expand up to 31K with this number of jobs being modelled.

These points caused a second version to be written and Fortran IV was selected for this for three reasons:

(1) Programs exceeding 32K could be written in this language but not in CSL.

(2) The "size" of the program would be finally fixed and known at compilation.

(3) This language is available on other machines.

The main aim of this new program was to reduce the run real time ratio to a more acceptable value. By keeping more of the job data in store it has been possible to reduce the disc file accessing to one per operation on a job, but this has extended the program to about 37K words; this is when modelling up to 3000 jobs in the machine shop at any one time. But this has reduced the time to simulate one day to under 10 minutes, although only after considerable programming effort.

For instance, with asynchronos timing there will be a cycle at every event and if, say, time is held to the nearest minute there could be a cycle through all the activities every minute thus causing many cycles and thus a slow program. But if the input data of operation times is rounded up to the nearest ten minutes, say, then the time must advance at least ten minutes each cycle thus reducing considerably the number of cycles and thus the time to simulate one day. Rounding up to the nearest ten minutes would seem to represent fairly well what actually happens as operatives rarely perform a complete operation, however short, in under this time.

The other major problem in slowing up this program came in deciding what job to take next from a machine queue. As originally modelled in CSL, all the jobs for this machine were in a special set. The CSL seemed only to consider for selection that special set of jobs, not examining all the others. But in the Fortran there was an array, the jth element of which held the number of the machine that job j was on. Thus for every machine if there was capacity on it, every job would have to be examined to see if it was on this machine. This meant m x n cycles through this activity for each cycle of the model where m = number of machines and n = number of jobs, and naturally this slows up the model. This was overcome by ordering all the jobs in machine order before the capacity for each machine is examined. Then this list is passed through only once for all the machines together.

Having the CSL program available was a great help in writing the Fortran program. In fact, it is doubtful whether the logic of the Fortran could have been made correct without the CSL to act as a guide.

## 6. PROGRAM DETAILS

In the computer used for these programs one word is 24 bits and 2 words are required for a floating point number. The program sizes and details are as shown in table 1, where the program numbers refer to the boxes in figure 2.

The whole of this system, starting completely from scratch, has been brought to this point with working programs by about 1½ man years of effort, excluding those concerned on data collection.

## 7. MODEL TESTS

Verification of the scheduler occurred as it went along since it was developed as a continual dialogue between the project team and production planning management. Establishing actual measures to compare the computer scheduler with that achieved by the clerical procedure for new batches was difficult, in particular as the clerks were not meticulous about sticking to the capacity restrictions. One measure used was the average number of operations scheduled per week on a particular set of data, and the number of batches that could not be loaded. Also production management looked at the schedules produced for new batches and commented on the suitability of the pattern suggested. To verify the rescheduler is almost impossible since this activity is not undertaken directly by clerical methods. The situation is illustrated by the solid lines in Figure 4.

In order to test the power of the rescheduler it has to be taken in conjunction with the simulation to forecast a flow of parts into finished part stores (dotted lines in figure 4).

This forecast can then be compared with what actually happens and what is desirable. But the simulation must be a very good representation of the machine shop if one is to make the statement: "If this rescheduler's plan is applied to the actual machine shop then this will be the result". Thus there is a need for a good simulation simply as a test bed for schedulers.

Verification of the simulation is harder than the verification of the scheduler since it does not produce a concrete plan which can be compared with actual clerical work, as new batches can, and also up to date there has been less dialogue in its development since the problems so far have been programming ones. The main measure for comparison is how the forecasted flow of parts into finished part stores compares with the actual. Other verification is mainly by observation, e.g., spending time at a machine centre to discover what priority rule is being used to decide on the next job and to compare what the simulation suggests should be at that centre with what actually is. The main function of the simulation is to forecast the flow of finished parts and to highlight the congestion points. The aim is to do this as realistically as possible by introducing into the model as little complication as possible. So if this can be achieved without, say, considering tooling requirements, it is not necessary to add these.

## 8. OPTIMISATION PROCEDURES

The scheduler developed in this case has certain parameters as input. The values for these parameters currently being used correspond to those used by the clerical procedure after modification as a result of discussion with the management. One use planned for this complete model is to find optimal values for these parameters with respect to some measure (e.g., average lateness, average queue lengths). Also, the way the data is sorted in box 3 of Figure 2 should be optimised.

Figure 5 indicates the way the complete model could be run. This can be done for several sets of parameters and several ways of sorting to find the optima.

Initial tests have with this system have given these results:

(1) Following a schedule produced after sorting on due date increased output by 100% over a two week period as compared to adopting a first come first served policy at each machine centre.

(2) Following a schedule produced after sorting on least slack per operation decreased output to 20% of that of following a due date plan over a two week period.

(3) Increasing capacity for bench work by 100% increased output by 60%.

From these results the following conclusions are suggested for this environment:

(1) It is worth while creating a schedule.

(2) A due date schedule produces better results over the short term than this least slack schedule.

(3) The small increase in labour and negligible capital required to increase bench work capacity can produce significant short term results.

In connection with point (2) in both lists above it is worth remarking on the difficulties in using least slack per operation in a heavily arrears situation. Least slack per operation is normally defined as

$$(d - t_o - \sum_{i=1}^{n} t_i)/n$$

where d = due time

$t_o$ = time now

$t_i$ = time for operation i

n = number of outstanding operations.

In a heavily arrears situation the numerator of this fraction is negative. Ordering by this value brought those jobs with few operations left and due dates only just in arrears ahead of those operations with many operations left and substantially in arrears. In order to redress this situation a large constant was added to the numerator before division in order to make it always positive. This then concentrated attention on those jobs with many outstanding operations almost regardless of their due dates.

A due date plan for a shop heavily in arrears will concentrate on arrears in the short term but may make little effort to correct this situation in the long term, whereas this is precisely the aim of the least slack plan used above. Which to use is a matter of management objectives.

The system in Figure 5 could be used to find a value for the constant to be added to the numerator in the fraction above which gave good results from the long and short term points of view.

## 9. CAPACITY SMOOTHING

Although the objective of the scheduler was not specifically to smooth capacity, this does seem to have been achieved well in all the runs done so far. Figure 6 is very typical, showing the hours committed by the scheduler's plan.

In almost every case, this extremely sharp cut off was noted, although it came in different weeks for different machines. This may arise purely because the environment is one of a considerable arrears situation.

## 10. OFFSHOOTS

The objectives of writing routines that could organise and control the passage of work through this job shop have been largely achieved by this work. But also there have been considerable offshoots on the way. Broadly speaking, these are:

(1) Modelling current procedures has helped management to understand the applications of the model and has led them to accept the results and understand what the model's limitations are. Thus they have become oriented towards a computer system and gained confidence in it.

(2) A system for rescheduling the machine shop has been produced and is already working, before the optimal values of the parameters are known.

(3) Improving this link in the control system has led to renewed effort in improving methods of reporting back what has been done in the machine shop, and in updating the component library with engineering changes. Thus improving just one part of a system has led to spontaneous improvements in adjacent parts since it is now seen that those improvements are of some use.

(4) Management are learning to ask new questions of the simulation.

One of the extraordinary effects of (2) is that now the operatives are receiving complete plans for work in weeks ahead, they are sticking more to what they should be doing since they feel someone is observing what happens. The progress manager now claims that the outputs from the scheduler are vital for retaining operative discipline!

## 11. SUMMARY

Simple clerical loading rules have been taken to form the basis for a tool to completely reschedule a machine shop. The origin of the rules and the continual dialogue with the management has given them confidence and experience in the use of computers. This method of dialogue has enabled systems that work but are not necessarily completely satisfactory to be produced very quickly.

The basic framework has been built in which to develop and test other methods of scheduling. One part of a large model of the factory has been built enabling management to see the effects of possible decisions.

The abstraction of the rules has produced a scheduler that is being applied in other job shops.

## BIOGRAPHY OF AUTHOR

C. E. Montagnon was educated at St. Pauls School in London and St. Johns College, Cambridge where he graduated in Mathematics. He served an apprenticeship with the General Electric Company Limited and then joined the Dunlop Company to specialise in the application of computers to management problems. He is currently studying for a Ph.D. at University of Warwick in the problems of scheduling in Production Control, while he is responsible for the applications of operations research techniques in the Engineering Group of the Dunlop Company at Coventry, England.

PROGRAM DETAILS

| Program | Size | Cards | MTs | Disc | Run Time |
|---|---|---|---|---|---|
| 1 | 12K | ✓ | 1 | 1 | 20 mins |
| 2 | 13K | ✓ | 2 | 2 | 30 mins |
| 3 | 8K | ✓ | 2 | 2 | 10 mins |
| 4 | 12K | ✓ | 2 | | 120 mins |
| 5 | 5K | ✓ | 1 | 1 | 15 mins |
| 6 | 37K | ✓ | 2 | 2 | 10 mins/1 day |

TABLE 1

OVERALL MODEL SCHEMATIC

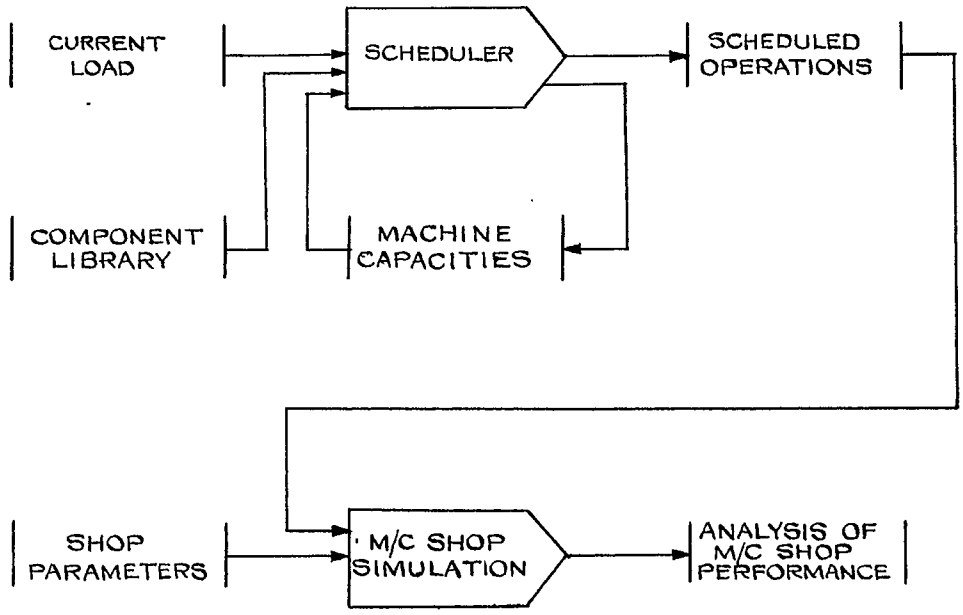


FIGURE 1

DETAILED MODEL SCHEMATIC

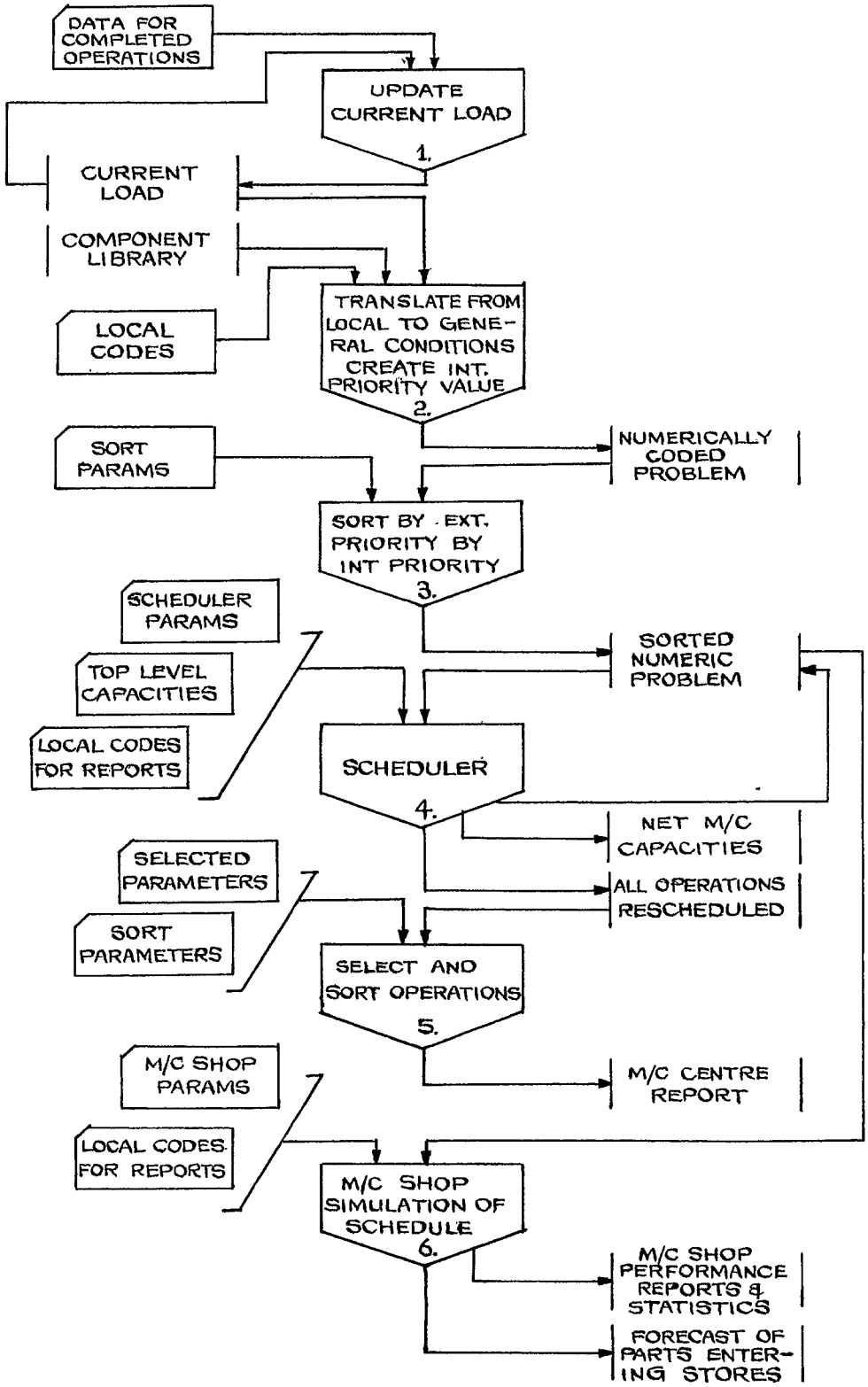


FIGURE 2

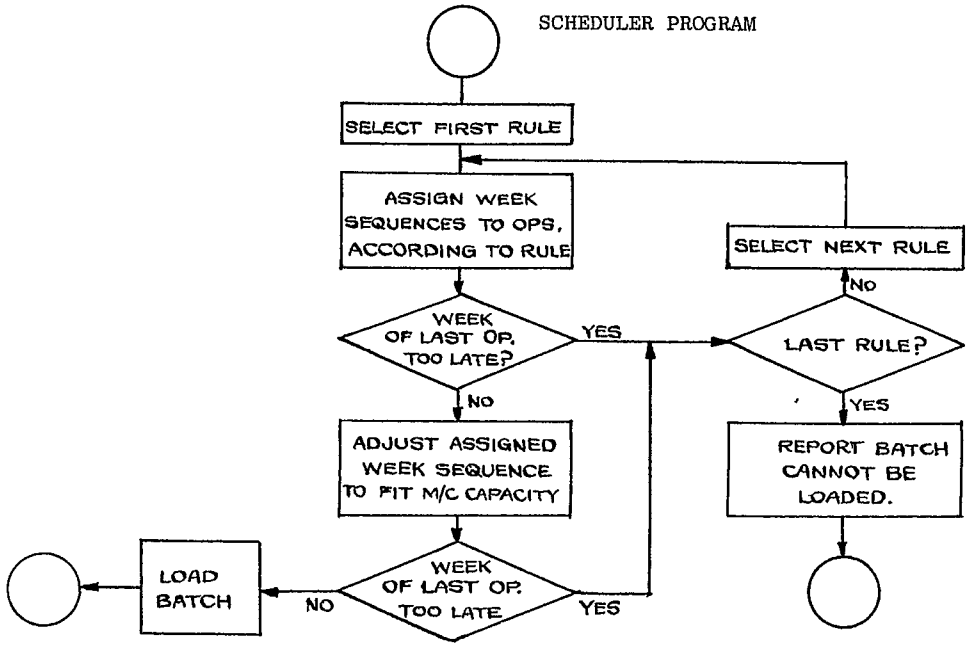


FIGURE 3

MODEL TESTS

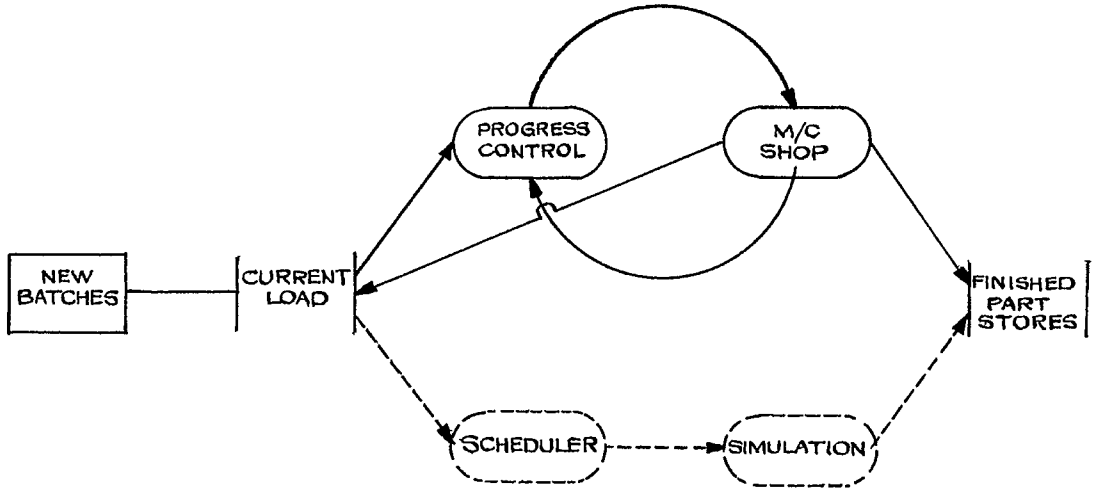


FIGURE 4

OPTIMISATION PROCEDURE

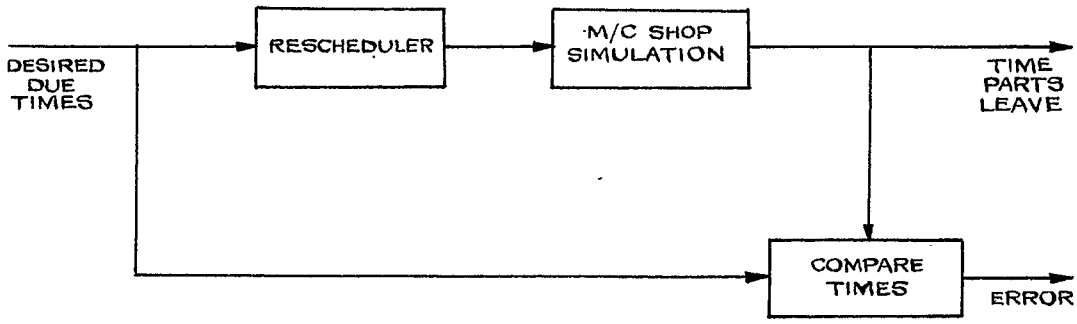


FIGURE 5

CAPACITY SMOOTHING

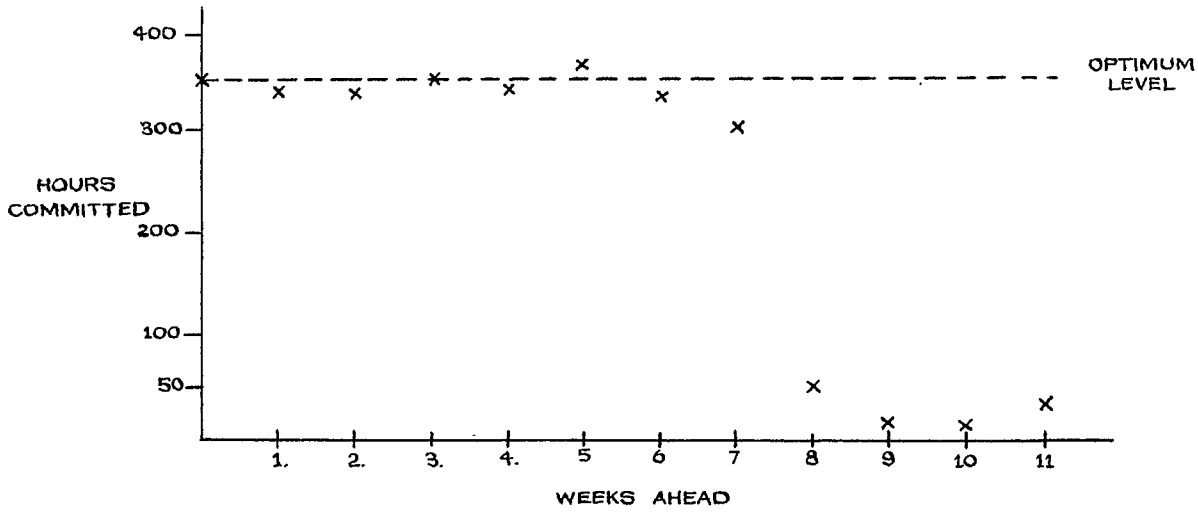


FIGURE 6