Thesis submitted for the degree of Doctor of Philosophy

"The Application of Computers to the Balancing and Sequencing of Assembly Lines"

by

J.L.C. Macaskill, B.Sc.(Eng.)

Department of Computing Science,
University of Adelaide.

8 December, 1969.

TABLE OF CONTENTS

APPENDICES

APPENDIX I

APPENDIX II

APPENDIX II (contd.)

LIST OF FIGURES

SUMMARY

The research reported in this thesis has been undertaken with the objective of adding to the understanding of mixed-model assembly. In the introduction, the nature of the problem is presented in general terms and the main work already done on it is reviewed.

Next, the assembly problem is theoretically formulated and the methods of assembly-line balancing are discussed in relation to the requirements of an effective assembly operatic It is shown that known methods have deficiencies and that the practical characteristics of the assembly line play a large part in determining which deficiencies are or are not acceptable.

The theoretical formulation is followed by a description of a computer program developed to investigate various aspects of assembly-line balancing. Considerable attention is given to economy in use of core store by the packing of precedence and other assembly information within the computer words. Various heuristic balance procedures are investigated and conclusions are reached that account for the relative effectiveness of certain of these procedures. Methods of achieving special requirements in assembly operations (e.g. allocation of special tasks to special stations) are included in the program and important implications of these procedures

are identified and discussed. It is concluded that there is a need for a proper understanding of these implications if computer techniques are to be applied effectively to the analysis and planning of assembly operations.

The investigation of balancing is followed by a description of a detailed computer simulation program prepared to study assembly-line performance. The inputs to this program are the allocations of work obtained from the balance program together with the mixed-model production requirement. This latter may be in the form of a specific sequence of product units or as a plain statement of how many are required of each model. The simulation is general enough to be applied to a wide range of assembly problems and permits variation of a wide range of parameters of the assembly.

The mathematical model on which the simulation is based includes algorithms for automatically generating acceptable sequences of product units both where no constraint is placed on the nature of the final sequence adopted, and where only a limited modification may be made to some given sequence of units. The effect of permitting operators to move out of their stations by different amounts upstream and downstream is examined, and the effect of permitting and excluding concurrency of operations when one operator is forced into another operator's station is investigated. The simulation program

has proved to be a powerful heuristic instrument and a variety of interesting conclusions are reached.

Next, the thesis reports the results of the application of the balance and simulator programs to a practical problem represented by a small self-contained assembly-line used for assembling front seats of motor cars. The results of this work showed the automatic sequencing algorithms of the simulation to be effective in setting up acceptable sequences in various situations. The balance program showed that significant economy in core store usage is obtained without severe penalty in computational speed.

The thesis ends with a discussion in which general conclusions are drawn from an overall consideration of the whole of the research and in which areas for further research are identified.

This thesis contains no material that has been accepted for the award of any other degree or diploma in any University, and to the best of my knowledge and belief contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

(J.L.C. MACASKILL)

..8. December, 1969.

ACKNOWLEDGEMENTS

I am greatly indebted to Professor R.B. Potts, Department of Mathematics, University of Adelaide, for much helpful advice and encouragement during the whole of the research.

I thank Professor J.A. Ovenstone and Dr. I.N. Capon, Department of Computing Science, University of Adelaide for their encouragement and support, and also thank the members of the Adelaide University Computing Centre for their help and tolerance.

All the practical material relating to assembly lines was obtained from General Motors-Holden's Pty. Ltd., and I acknowledge with gratitude the assistance and advice given me by Mr. Bruce Black and many other officials of the Woodville and Elizabeth plants of General Motors-Holden's Pty. Ltd., South Australia.

I greatly appreciated discussions of my work with Mr. E.M. Mansoor of the University of Melbourne, and thank him for valuable comments made.

CHAPTER 1. INTRODUCTION

1.1 Historical background

Traditionally, assembly lines have been used to assemble large numbers of exact copies of a given product. The principle used is to carry a sequence of products past equally spaced operators at constant speed. Each operator must execute a predetermined set of tasks in the period while a product unit is moving past a zone allocated to him and called his station. The fundamental problem of the assembly line thus defined is the allocation of work to operators in shares that are as equal as possible. The allocation is constrained by rules that require certain tasks of the assembly to be done before others and by the need for tasks not to be subdivided beyond some predetermined level.

Consideration of any familiar small assembly operation (e.g. changing the wheel of a car) illustrates that many different combinations of the tasks of the assembly would achieve an acceptable end result. Consideration of the difficulty of sharing out the tasks in the wheel example given above equally between say, four people, illustrates the difficulty of the allocation.

One method of allocating is to identify every feasible combination of tasks, where a feasible combination is a

combination that results in correct assembly of the product without any task requiring to be executed more than once. From this set of combinations the best is chosen or, in case of a tie, one of the best, and the result is an optimal solution. The difficulty is that there are often several million feasible combinations in a practical assembly operation and the method fails through the sheer volume of the computation.

One of the first mathematical formulations of the assembly-line balancing problem was given by Salveson [18] who proposed a linear programming solution which was too arduous to be practicable. Jackson [7] proposed a dynamic programming solution in 1956 and Held et al. [3] proposed another dynamic programming solution in 1963. Both these methods were able to identify the best solution but on account of the heavy computational load were not suitable for practical assembly lines. Held et al. [ibid.] also proposed a dynamic programming method that although it was unable to identify the optimal solution, nevertheless was able to produce near optimal solutions for full-scale assembly operations.

Various heuristic methods designed to produce an acceptable solution quickly were produced during the early

and middle sixties. In most instances these methods were well suited to automatic-computer solutions. Examples of the method are Kilbridge and Wester [9], Helgeson and Birnie [4], Moodie and Young [15], and Mansoor [13]. In the latter two references more complex second stage procedures were presented that allowed near optimal and optimal solutions to be obtained at the cost of an additional computing load. An interesting review of the earlier work was given by Ignall [6] and an informative discussion of the general problem posed by the assembly line was presented by Kilbridge and Wester [10] in 1961.

Tonge [21] in 1965 presented a method based on a probabilistic combination of heuristics, while Arcus [1] proposed an interesting and effective method in which a large number of feasible solutions were generated by a random number technique and the best result was selected. Buffa [2] reported experiments by A.A. Mastor in which the various methods were examined and compared. Later work on balancing mainly concerned with modification of algorithms and improvement of techniques was presented by Mansoor and Yadin [14] and Heskiaoff and Weinstein [5].

Within the last few years, prominence has been given to the problem of making different models of the same general

product on one assembly line. This poses two main problems. First the work of the different models requires to be allocated to the stations. Methods are proposed for this by Thomopoulos [19,20] and Macaskill [12]. Arcus [1] discusses this problem briefly. Then, having allocated the work, it is necessary to consider the sequencing of the product units onto the assembly line. Because of the different work content of the different models an ill-conditioned sequence can severely degrade assembly performance. Aspects of this problem are considered by Wester and Kilbridge [22], Thomopoulos [ibid.], and briefly by Arcus [ibid.].

1.2 Method of presentation

In this thesis the problems of balancing and of sequencing for mixed-model assembly are examined and computer techniques are developed for their solution. The material of the thesis is presented as follows.

In Chapter 2 fundamental aspects of the problem are considered. First, relevant elements of graph theory are presented and important features of single-model assembly are summarised. This leads on to two characterizations of the mixed-model assembly line. The mixed-model balance problem is then formulated and the philosophy of the use

of assembly lines in production processes is discussed. Finally two alternative methods of solution of the mixed-model assembly problem are described and discussed.

Chapter 3 is concerned with practical considerations of the balance problem and with its solution by automatic computation. Matters in single-model balancing that relate to mixed-model balancing are discussed and the two methods of solving the mixed-model balance presented in Chapter 2 are dealt with in more detail. Heuristic balance procedures are described and explained and material of interest in the development of the computer program used here for balancing are presented. Aspects of single-model balancing that relate to mixed-model work are investigated and the results of computer running are given and discussed. Also, results obtained with one of the proposed methods of mixed-model balancing are considered, and methods for achieving special balance requirements are described.

Chapter 4 is concerned with the computer program that has been developed for simulation of assembly line performance. The problem to be solved by the simulation is given, and the mathematical model is defined. The method of application of this model to the computer is explained, and algorithms for automatically generating suitable sequences of units in various conditions are

presented. Simulator test running is described and the results are analysed. A considerable number of interesting conclusions are reached.

In Chapter 5 the balance and simulation computer programs are applied to a practical assembly-line problem. The results of a number of computer runs are reported and analysed, and conclusions are reached concerning assembly line behaviour and performance of the sequencing algorithms. Results concerning the allocation of tasks that require simultaneous work by two operators are discussed.

Chapter 6 presents a discussion of the major conclusions of the research. The conclusions are presented under the headings concurrency of work, prediction of assembly performance, speed and practicability of computations and the organization required for the planning of assembly operations. It is considered that the findings given under these headings have a wide application and are of considerable interest.

CHAPTER 2. THEORETICAL FORMULATION

The analysis of an assembly operation can best be formulated mathematically by using some of the concepts of graph theory [8]. Much of the notation and terminology used in discussion of mixed-model assembly is based on that used by Reiter in a recent presentation of the characterization of single-model assembly lines [17].

2.1 Elements of graph theory

A directed graph $G = (N,L)$ is defined algebraically as a finite non-empty set N of unordered elements and a finite set L whose elements are ordered pairs of elements of N. The elements of N are called nodes and are denoted by i or n_i, i=1,2,...,g. The elements of L are called arcs and are denoted by l_h,h = 1,2,...,f. We write $l_h = (n_i,n_k)$ or $l_h = (i,k)$ signifying the arc with initial node n_i or i and terminal node n_k or k. A node that is neither an initial node nor a terminal node is called an isolated node.

For the application of graph theory considered in this thesis it is important for the elements n_i and n_k defining an arc to be distinct. Any reference to a graph in this thesis therefore carries with it this implication. Despite this exclusion it is useful to note that an arc with initial node n_i the same as its terminal node n_k is called a loop.

If $n_1, n_2, \ldots, n_g$ are distinct nodes of G such that $(n_i, n_{i+1}) \in L$, $i=1,2,\ldots g-1$, then the sequence

$$n_1, (n_1, n_2), n_2, \ldots\ldots n_i, (n_i, n_{i+1}), n_{i+1}, \ldots\ldots, n_g$$

defines a chain of length (g-1) from node n_1 to node n_g. This chain can be described unambiguously either by the sequence of nodes $n_1, n_2 \ldots, n_g$ or by the sequence of arcs $(n_1, n_2), (n_2, n_3), \ldots (n_{g-1}, n_g)$. An arc (n_i, n_k) is a chain of length 1. An arc (n_i, n_k) is said to be redundant if there exists in addition to the arc itself a chain from n_i to n_k.

A cycle is a chain except that $n_g = n_1$. A graph that contains no cycles is called acyclic.

The terms elemental task and assembly operation will be used frequently in the material that follows. An elemental task is a member of the set of the economic subdivisions of the work entailed in assembling some product. An assembly operation is a set of elemental tasks so ordered and selected that all can be completed. Such a set of elemental tasks is called feasible. An assembly line will be regarded as continuously in operation from start to finish of a production period or shift. Typically, this period would be of about 8 hours duration.

2.2 Single-model assembly line

Before developing a graph-theoretic representation of a mixed-model assembly line, it is helpful to summarise certain aspects of Reiter's representation [17] for a single-model assembly line, as follows:-

A single-model assembly line is characterized by an acyclic directed graph G with nodes $N = \{n_1, n_2, \ldots, n_g\}$ and arcs $L = \{l_1, l_2, \ldots, l_f\}$, together with a set of durations $t = \{t_1, t_2, \ldots, t_g\}$ where $t_i > 0$ is associated with node n_i. The presence in G of a node n_i indicates that an assembly task i of duration t_i is part of the assembly operation characterized by G. Also, the presence in G of an arc l_h joining n_i to n_k indicates that task i must be completed before task k can start. We write $l_h = (n_i, n_k)$ and $n_i \rightarrow n_k$ or, alternatively, $i \rightarrow k$. The concept associated with the notation $i \rightarrow k$ is called a precedence relation. If $\gamma(i)$ represents the time at which i is started, then $i \rightarrow k$ implies

$$\gamma(i) + t_i \leq \gamma(k) \qquad \ldots (1)$$

A single-model assembly line defines the precedence relations in assembly operations that permit manufacture of identical copies of a given product model in any quantities required. The relation (1) specifies that in

the assembly of a particular product unit the ith task must be complete before the kth may start. A further constraint may be placed on the assembly operation by requiring that there shall be no concurrency of tasks in a particular product unit irrespective of whether or not precedence relations apply in the instance concerned. Concurrency of tasks takes place when two tasks in one and the same product unit are being carried out simultaneously by separate agencies. Constraint on concurrency of tasks plays an important part in assembly line behaviour and is discussed further later in this Chapter.

2.3 Mixed-model assembly line

A mixed-model assembly line defines the precedence relations in an assembly operation that permit manufacture of a number of sets of identical copies of different product models. In general such models will have some tasks in common. This leads to an obvious multigraph representation as follows.

2.3.1 Multigraph characterization

A set of assembly operations associated with models $M = \{1,2,\ldots,p\}$ is characterized by a set of acyclic directed graphs $S = \{G_1, G_2, \ldots, G_p\}$. Each graph G_m is associated with model m, has nodes $N(m) = \{n(m)_1, n(m)_2, \ldots$

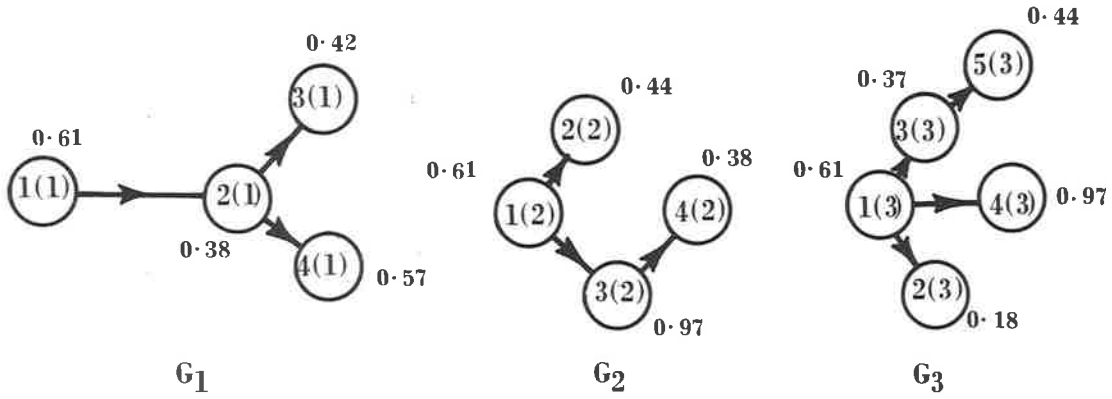
and arcs $L(m) = \{l(m)_1, l(m)_2, \ldots\}$ together with a set of durations $t(m) = \{t(m)_1, t(m)_2, \ldots\}$ where $t(m)_i > 0$ is associated with node $n(m)_i$. In corresponding manner to the single-model graph G we take $l(m)_h = (n(m)_i, n(m)_k)$ to indicate that $n(m)_i \rightarrow n(m)_k$. We write $i(m) \rightarrow k(m)$ to signify that task i of model m precedes task k of model m.

The graphs G_1, G_2, G_3 shown in figure 1a are called precedence graphs or diagrams and characterize assembly operations for models 1, 2, 3. To illustrate the notation used, in G_1 the node labelled 3(1) has the number 0.42 associated with it. This signifies that task 3 of model 1 has a duration 0.42 time units. Certain nodes have equal durations (e.g. 2(2) and 5(3)). For the purposes of the following example assume that when two nodes have equal duration they both relate to the same task (e.g. the tasks 2(2) and 5(3) are the same task).

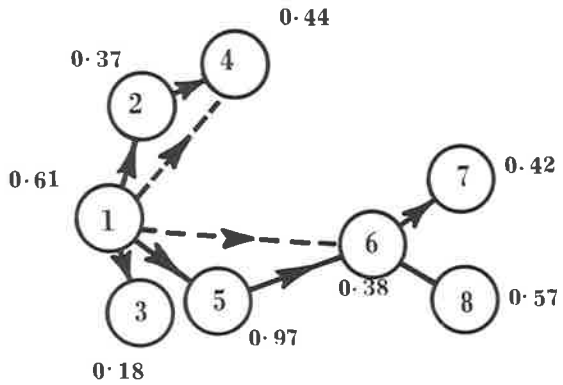
2.3.2 Characterization by combined graph

A combined precedence graph G_M that characterizes the set of models $M = \{1,2,\ldots p\}$ may be obtained from the graphs $G_1, G_2, \ldots, G_p$ as follows:-

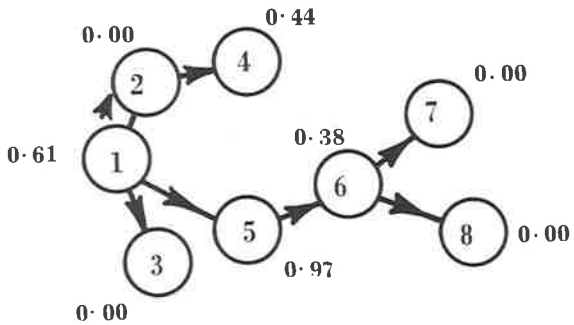
(i) Form the set of nodes N(M) from the sets of nodes $N(1), N(2), \ldots, N(p)$ associated with models



(a) Precedence graphs for models 1, 2 and 3



(b) Combined precedence graph



(c) Equivalent precedence graph for model 2

FIGURE 1 : DEVELOPMENT OF COMBINED PRECEDENCE GRAPH

1, 2,...p, as follows

$$N(M) = N(1) \cup N(2) \cup ..... \cup N(p)$$

(ii) Form the set of arcs L(M) from the sets of arcs L(1),L(2),...,L(p)

$$L(M) = L(1) \cup L(2) \cup ..... \cup L(p)$$

Redundant arcs in G_M may be omitted.

Figure 1b shows the combined graph G_M obtained from G_1, G_2, G_3 of figure 1a. The nodes of G_M have been numbered arbitrarily. Arcs (1,4) and (1,6) are redundant and may be omitted. Node 6 of G_M, node 2(1) of G_1, and node 4(2) of G_2 all relate to the same task. The duration (0.38 time units) of task 2(1) is therefore the duration associated in figure 1b with task 6.

In order to identify whether node $n(M)_i$ of G_M represents a task of model m, a vector V(i) is associated with every node $n(M)_i$ and is called a model identification vector, where

$$V(i) = (v_{im} | m = 1,2,...,p)$$

If model m does not contain task $n(M)_i$ then $v_{im} = 0$: if it does contain this task, then $v_{im} = 1$. Let the duration associated with every node $n(M)_i$ in G_M be multiplied by the mth component of the vector V(i), and let the result of the multiplication be associated with the relevant node

$n(M)_i$. Then, a new graph G'_m will be obtained that is equivalent to G_m, one of the graphs from which G_M was formed.

As an example, from figures 1a and 1b we have $V(6) = (1,1,0)$ and graph G'_2 in figure 1c is equivalent to G_2 of figure 1a.

This second characterization is well suited to the attainment of savings in usage of rapid access storage in computer assembly-line-balance problems.

2.4 The assembly-line balance problem

In the balancing of an assembly line, the assembly tasks are apportioned to work stations manned by one or more assembly workers. The product is transported past these stations in a way that facilitates the requirements for its assembly. The mixed-model balance problem is formulated below.

2.4.1 The mixed-model balance problem

Let set H have members S(m), where S(m) corresponds to the mth product model. Let each member S(m) be a set of sequences $B(m)_1$, $B(m)_2$,..., where the number of members need not be the same in each set of sequences. Let each task i of each model m occur exactly once in one and only one sequence $B(m)_j$ of the mth set of sequences S(m).

If the sets S(m) are to be feasible there must exist a situation such that if

(a) $B(m)_j = (n(m)_{j1}, n(m)_{j2}, \ldots, n(m)_{jf})$, ... (2)

then,

$$\gamma(n(m)_{jk}) + t(m)_{jk} \leq \gamma(n(m)_{j,k+1}), \quad \ldots (3)$$

where $k = 1,2,\ldots,f-1$

and $m = 1,2,\ldots,p$

where f is the final node of each sequence $B(m)_j$ and where p is the number of models.

(b) $B(m)_j = (n(m)_{j1}, n(m)_{j2}, \ldots, n(m)_{jg})$ ... (4)

and

$$B(m)_{j+1} = (n(m)_{j+1,1}, n(m)_{j+1,2}, \ldots, n(m)_{j+1,h}) \quad \ldots (5)$$

then

$$\gamma(n(m)_{jg}) + t(m)_{jg} \leq \gamma(n(m)_{j+1,1}) \quad \ldots (6)$$

The sequence $B(m)_j$ represents the ordered set of tasks of model m that are done in assembly station j. Condition (a) specifies that for every unit of each model the order of the tasks in $B(m)_j$ shall be consistent with the precedence relations. Condition (b) requires that for every unit of each model all tasks in a given station must be complete before the first task in the succeeding station is started, or in other words, that there shall be no

concurrence of operations in different stations.*

The work content w_m of model m is given by

$$w_m = \sum_{i=1}^{q_m} t(m)_i$$

where q_m is the number of elemental tasks in model m. A production period is called a shift and has duration T time units. Let f(m) be the number of units of model m required per shift and let p be the number of different models required. Then the total work content W_T of the shift is given by

$$W_T = \sum_{m=1}^{p} f(m)w_m \qquad \ldots (7)$$

Let λ be the number of assembly stations needed to meet the production requirement of the shift and let the total effort supplied for the shift be defined as λT, the total potential for doing work during the shift. Then, the operator idle time I accumulated during the shift is given by

$$I = \lambda T - W_T \qquad \ldots (8)$$

---

* Note that although according to condition (b) above no two operators in different stations may work simultaneously on the same product unit, there is no objection to concurrent operations on different product units. Indeed, this concurrence of work on different units is one of the key features of the assembly line.

The mixed-model balance problem may now be formulated as follows: determine feasible sets S(m) that permit a production requirement f(m) to be met for each model m and that minimise the idle time I. A perfect solution is achieved when $W_T = \lambda T$.

2.4.2 Assembly-line philosophy

Before considering methods of approach to the balance problem, discussion of the practical philosophy of the assembly line is given. It will be seen that no solution method for the mixed-model assembly problem will fully satisfy all the requirements of a practical assembly line.

In a practical assembly line products are transported from worker to worker. The intention is to allocate the work in such a way that each worker may complete his allocation before each product leaves an area in which it is permissible for him to do his work. The main advantages conferred by the use of the assembly line are:-

(i) The efforts of many workers of different skills can be applied efficiently to a single unit of a product. As a corollary, much advantage will be lost if workers are allocated tasks in which they are not skilled. The regularity with which a given type of work can be allocated to a given worker will affect the investment needed in the

training of workers for their tasks.

(ii) Each worker can be given room to do his work unimpeded by other workers. Tools and materials can be placed to suit the convenience of the worker, and hence to reduce unproductive work entailed in their movement to the job and away from it.

(iii) Work can be shared out fairly between workers, and by proper allocation of tasks worker idle time can be kept within bounds.

(iv) The passage of products at a constant speed facilitates the maintenance of an agreed rate of working. This has the double advantage of keeping up an acceptable production rate and of facilitating the planning of future work.

2.4.3 Mixed-model balance by multiple individual balances (method 1)

The first approach to the mixed-model balance is suggested by the multigraph characterization given in 2.3.1 above. This entails the execution of individual balances for each model.

The theoretical aspects of single-model balances are suitably presented by Salveson [18], and particular matters

that have relevance here are now presented.

It is required that the rate of transport of product units shall be constant throughout the shift and the basic principle of the balance is the apportionment to assembly stations of amounts of work that are as nearly as possible equal to each other. Let c_j be the duration of tasks allocated to station j. The quantity c where

$$c = \max_j c_j \qquad \ldots (9)$$

is called the cycle time. In order that no worker need leave his station to complete his task, the time taken for each unit to pass through each station is set equal to c and is here called the station passage time. In these conditions it can be shown that the interval between application of units to the assembly line should also be set equal to c. This interval is called the product launch interval. For present purposes it is assumed that there will be one operator per station giving an operator effort per station per product unit of c operator time units.

In a development of the single-model balance to cater for the mixed-model situation [Wester and Kilbridge, 22] it is assumed that the work required can be evenly divided between the stations. That is, in theory, we may write

$$c(m)_j = c(m),\ j=1,2,\ldots,\lambda \qquad \ldots (10)$$

where $c(m)_j$ is the work required to complete the tasks allocated to station j for model m, and c(m) is the effort available at each station for model m. λ is the number of stations in the line.

The work allocated to each station for model m will, in practice be less than or equal to the effort c(m) supplied at the station and will, in general, approximate closely to c(m). c(m) is called the model cycle time of model m.

If f(m) is the number of units of model m required, the effort made available at each station for all the units of model m is c(m) f(m), and the total effort available at each station during one shift is given by

$$\sum_{m=1}^{P} c(m)\, f(m)$$

If units are launched at equal intervals*, the minimum launch interval β that will guarantee that work required does not exceed effort supplied is given by

---

* The solution by launching at variable intervals is considered by Wester and Kilbridge [22] as well as fixed interval launching. Consideration of variable interval launch is purposely omitted here because it is considered that, in general, it presents so many practical difficulties that it has no place in a general treatment. In particular instances the method may be of considerable value.

$$\beta = \sum_{m=1}^{P} c(m)\, f(m) \;/\; \sum_{m=1}^{P} f(m) \qquad \ldots (11)$$

This relation shows that if $c(m)_{max}$ and $c(m)_{min}$ are the maximum and minimum values of c(m) and if $c(m)_{max} > c_m(min)$ then

$$c(m)_{max} > \beta > c(m)_{min}, \qquad \ldots (12)$$

where β is called the production cycle time.

2.4.4 Discussion of method 1

The foregoing abbreviated treatment is sufficient to demonstrate the two serious weaknesses of method 1 as follows:-

(i) Consider, as an example, a 4 station mixed-model assembly line. Model1 is assumed to contain tasks {1,2,3,4,5,6,7,8} allocated evenly to stations as follows:-

Station 1(1,2), station 2(3,4), station 3(5,6), station 4(7,8).

If model 2 contains tasks {3,4,5,6}, the allocations, again made evenly, might be

station 1 (3), station 2 (4), station 3 (5), station 4 (6).

Because of the even distribution the work in stations 1 and 4 for model 2 has nothing in

common with the work in stations 1 and 4 for model 1.

Such a distribution is likely to break the rule that workers should be allocated tasks in which they are skilled.

(ii) Operator idle time occurs from the instant when an operator has finished his tasks on one unit until work becomes available on the next unit. Congestion occurs from the instant when in order to finish his tasks an operator is forced to leave his station until these tasks are finished or work on the unit is terminated. Operator idle time and congestion can easily occur with balance method 1. Consider, as an example, a situation in which model p is the model with greatest work content, and let b units of this model reach the line in immediate succession. Let entry time of the first model be zero and consider station 1.

Duration of work on b units of model p = $bc(p)$

Time of entry of unit (b+1) to line = $b\beta$

The period t_D between completion of work on unit b and entry of unit b+1 is given by

$$t_D = bc(p) - b\beta$$

If $c(p)$ = 3 minutes, β = 2 minutes (not unreasonable values) and b = 10, then

$$t_D = 10 \text{ minutes.}$$

In other words, because of congestion, the operator in station 1 cannot start work on unit (b+1) until 10 minutes after it has entered his station. The situation is ridiculous and is caused by an unsatisfactory sequence of models on to the line. Corresponding difficulties arise when an unrelieved sequence of models with model cycle times significantly less than the launch interval is applied to the line.

2.4.5 Solution by aggregated task-group allocation (method 2

Consider figure 1b and let the sums A_i be formed where

$$A_i = \sum_{m=1}^{p} t(M)_i v_{im} f(m); \quad i=1,2,\ldots,q \qquad \ldots (13)$$

where q is the number of nodes in the combined precedence graph G_M and where v_{im} is the mth component of the vector V(i) associated with node i (see Section 2.3.2 above). A_i is interpreted as the aggregated duration of every repetition of the elemental task i for every unit in the model mix

during the shift and is called the task group duration of i. Let F be a set of sequences $B_1, B_2, \ldots, B_\lambda$ of nodes n_i of the set N(M) such that each n_i occurs exactly once in one and only one B_j. Consider the feasibility of the B_j, where

$$B_j = (n_{j1}, n_{j2}, \ldots, n_{jg}) \qquad \ldots (14)$$

B_j is a sequence of nodes of the combined precedence graph and has associated with it p sequences B_{jm}, one for each model in the mix where, for example,

$$B_{jm} = n_{j1}v_{j1,m}, n_{j2}v_{j2,m}, \ldots, n_{jg}v_{jg,m} \qquad \ldots (15)$$

Any sequence of tasks present in B_j will at some time during the shift be associated with every model in the mix in the form shown in eq. (15) above. The sequences B_{jm} will be feasible only if the conditions given in eq. (2) to (6) of 2.4.1 are met. In practice these conditions are met if the precedence relations of the graphs G_m are observed. Therefore if the sequences B_j are to be feasible they must be such that none of the precedence relations G_m are broken. But the precedence relations of the G_m are all contained within the combined precedence graph G_M.

Therefore, if the B_j are consistent with the graph G_M they will be feasible. The B_j will correspond to an

ordered set of tasks allocated to station j. Further, if nodes are allocated to station j as in eq. (14) above, then associated with each node n_{jk} there will be an elemental task duration $t(M)_{jk}$ and an a task group duration A_{jk} where

$$A_{jk} = \sum_{m=1}^{p} t(M)_{jk} v_{jk,m} f(m) \qquad \ldots (16)$$

and from eq. (16) we obtain the total amount of work W_j' allocated for the whole shift to station j, thus

$$W_j' = \sum_{k=1}^{g} A_{jk}$$

where g is the last node in the sequence B_j. Since no operator can during the shift do more work than one operator-shift we have as the second condition of the aggregated task group balance

$$T \geq W_j', \quad j = 1,2,\ldots,\lambda.$$

The balance problem of method 2 is then as follows: determine a set F of feasible sequences B_j such that λ is minimized and such that the total duration of the work W_j' allocated to station j nowhere exceeds the shift duration T.

2.4.6 Discussion of method 2

Method 2, balancing by aggregated task groups, has the advantage that if one elemental task i is allocated to station j, then so is every other elemental task i that has to be done to assemble the whole model-mix. The allocation of tasks to those trained to do them is therefore greatly facilitated.

The method also has distinct disadvantages. Although the work represented as a sum of task groups is shared evenly between the stations, the tasks associated with a given model will not, in general, be shared between stations in the same even way. For example, in the illustration given in section 2.4.4 it was assumed that model 1 of a model-mix contained tasks 1 to 8 balanced over four stations thus:-

1 (1,2), 2 (3,4), 3 (5,6), 4 (7,8).

The second model was assumed to contain tasks {3,4,5,6}. The allocation of these four tasks obtained with method 2 would necessarily be

1 (0), 2 (3,4), 3 (5,6), 4 (0)

resulting in an exceedingly uneven distribution of work between stations for model 2.

In practice, in exacting conditions, a severely uneven

distribution of work may occur for some particular model. If nothing can be done to modify this distribution, then it may be necessary to increase the length of assembly line allocated to certain stations. In extreme instances a station passage time of three times the launch interval might be required in one station. This has the immediate disadvantage that the length of assembly line (and size of in-process inventory) required may be much greater than that required for method 1 above. There is the further disadvantage that if the line has station equipment of low mobility and if the model-mix is changed frequently, severe difficulties may be met in properly locating the station equipment.

Method 2 will also suffer from the same disadvantage as method 1 in that the assembly line will be sensitive in its behaviour to the sequence in which the units are applied to the line. In fact, in general, the station passage times will, in method 2, differ much more widely from the launch interval than will be the case in method 1, and as a result sensitivity to launch sequence will be more evident in method 2 than in method 1.

2.5 Concluding remarks

The discussion of the two suggested balance methods

shows that the choice offered is one that will be decided from practical considerations. It is possible to visualise situations in which either method might be acceptable or unacceptable. Both methods are therefore included in the work described in Chapter 3.

In the formulation of the balance problem, the possibility of relaxing the constraints on the problem by permitting concurrent work in stations has not been included in any of the balance methods. However, in the simulator runs situations are investigated both where concurrent work is permitted and where it is excluded.

CHAPTER 3. COMPUTER BALANCE-PROGRAM

3.1 General

The problems of single-model and mixed-model balances have been formulated in Chapter 2. Chapter 3 is concerned with the practical aspects of applying the problem as formulated to a computer and with the quality and nature of the solutions obtained. It was shown in Chapter 2 that both variations of the formulation of the mixed-model balance were closely associated with the single-model problem. Practical aspects of the formulation of this latter problem are therefore presented now, and lead on to the discussion of the practical aspects of mixed-model balances.

The discussion of single-model balancing and of heuristic procedures that follows relies for its material on many sources. The following references however, in particular, have a general relevance to the material presented. Salveson [18], Helgeson and Birnie [4], Moodie and Young [15], Mansoor [14], and Kilbridge and Wester [10]. Throughout section 3.2 below it is assumed that each assembly station is manned by a single operator.

3.2 Single-model balancing, practical considerations

3.2.1 Iterative method

An assembly situation is as follows. The duration of

the ith elemental task of a single-model product that has k tasks is t_i. Shift duration is T; production requirement is P units per shift. The maximum value c_{max} of cycle time that will permit the requirement to be met is

$$c_{max} = T/P \qquad \ldots (1)$$

Let λ be the number of stations required. Knowledge of c_{max} permits definition of λ_{min}, the minimum number of stations by which the production requirement could theoretically be obtained, as follows

$$\lambda_{min} = \sum_{i=1}^{k} t_i/c_{max}, \quad \text{where } \sum_{i=1}^{k} t_i/c_{max} \text{ is an integer}$$

$$\text{or} \quad \lambda_{min} = \left[\sum_{i=1}^{k} t_i/c_{max} \right] + 1 \text{ , otherwise}$$

where the square bracket denotes that the non-integer part of its contents is to be discarded. Having obtained λ_{min}, the minimum theoretical cycle time c_{min} is given by

$$c_{min} = \sum_{i=1}^{k} t_i/\lambda_{min} \qquad \ldots (2)$$

As an example let T = 480 minutes, P = 240, then

$$c_{max} = 2 \text{ min.}$$

Let k = 10 and let the 10 values of t_i be

{0.5, 0.7, 1.1, 1.0, 0.4, 1.5, 1.2, 0.8, 0.4, 0.9}

then $$\sum_{i=1}^{10} t_i = 8.5$$

and $$\lambda_{min} = [\frac{8.5}{2}] + 1 = 5$$

and $$c_{min} = \frac{8.5}{5} = 1.7 \text{ min.}$$

If a balance could be obtained with $c = 1.7$ and $\lambda = 5$, there would be no operator idle time. Such a result is unlikely but not impossible, and therefore the first attempt at a balance might be made with $c = c_{min}$.

Assume that a method ξ has been defined in which elemental tasks will be allocated to stations in such a way that precedence constraints are observed and that the sum of allocated tasks does not exceed the chosen value c of cycle time. An effective approach to the balance would then be as follows.

(i) Attempt to balance by applying the method ξ with cycle time equal to c_{min} and with λ_{min} stations. If the balance fails go to (ii).

(ii) Let $c = c_{min} + \delta c$ where δc is an arbitrarily chosen small increment of cycle time. Attempt again to balance to c. If the balance fails then set $c = c_{min} + 2\delta c$ and continue with balance attempts and increments either until a balance

is achieved or until the incremented value of c exceeds c_{max}.

(iii) If the cycle time exceeds c_{max} the production requirement cannot be met (see eq. (1) above). Since a balance cannot be achieved with λ_{min} stations with $c \leq c_{max}$, the value of λ must be increased if a balance is to be achieved therefore let

$$\lambda = \lambda_{min} + 1$$

and determine the new minimum value of c, c'_{min}, that corresponds to this new value of λ as follows

$$c'_{min} = \sum_{i=1}^{k} t_i/(\lambda_{min} + 1) \qquad \ldots (3)$$

(iv) Proceed as in (ii) above with c'_{min} in place of c_{min} either until a balance is achieved or c exceeds c_{max}, when the number of stations must again be incremented as in (iii).

Various balance methods ξ are discussed later in this Chapter. However, irrespective of the balance method chosen, a balance will eventually be achieved by iteration in the manner described above. Let overall balance efficiency be defined as the ratio of work done to effort supplied and

denoted by ε. Then ε is given by

$$\varepsilon = \sum_{i=1}^{k} t_{i/\lambda c} \qquad \ldots (4)$$

Note, in iterative balances, that if the final value of c is less than c_{max}, the shift production will be given by T/c which exceeds the requirement P by a number of units given by

$$T\left(\frac{1}{c} - \frac{1}{c_{max}}\right) \qquad \ldots (5)$$

This extra production may not be required and, indeed, may prove to be an embarrassment. Curtailment of the shift duration may also not be practicable, and in a situation of this sort the single-pass approach described below may be helpful.

3.2.2 Single-pass method

The notation used is that of 3.2.1 above. Apply the balance method ξ with cycle time equal to c_{max} where, as in eq. (1) above,

$$c_{max} = T/P$$

but with the number of stations undefined. Continue allocating tasks to stations until no more tasks remain to be allocated. The number of stations λ required in practice is then given by the number required to complete the

balance. The method has the advantage that the balance is achieved in a single pass with the same number of stations as would be achieved by the method ξ used with iterative balancing. The disadvantage is that the operator in the final station may be grossly underemployed. To illustrate this consider a simple example where:-

$P = 480$ units, $T = 480$ min., $k = 6$

$t_1 = 0.87$ min., $t_2 = 0.60$ min., $t_3 = 0.40$ min.,

$t_4 = 0.41$ min., $t_5 = 0.56$ min., $t_6 = 0.29$ min.

Task 1 precedes tasks 2, 3 and 4, task 4 precedes tasks 5 and 6. From eq. (1), $c_{max} = 1.00$. A feasible allocation is as follows:-

Station	Tasks allocated (min.)	Residue of effort (man min.)
1	1 (0.87)	0.13
2	2 (0.60),3 (0.40)	0.00
3	4 (0.41),5 (0.56)	0.03
4	6 (0.29)	0.71

The unused residue of effort in station 4 is 0.71 man min. There may be difficulty in ensuring that this unused effort is not wasted. Balance overall efficiency as defined in eq. (4) above in such circumstances tends to reflect experience in the final station rather than

excellence of balance method. This latter is better represented by a criterionhere called the subterminal efficiency of the balance ε' where

$$\varepsilon' = \frac{\text{sum of tasks allocated to first } (\lambda-1) \text{ stations}}{c(\lambda-1)} \quad \ldots (5)$$

3.3 Mixed-model balancing, practical considerations

Two methods for the mixed-model balance problem were introduced in Chapter 2. These are now considered, starting with method 2.

3.3.1 Allocation by aggregated task groups

Comparison of material on single-model balances just presented with the mixed-model balance given in section 2.4.5 above shows a correspondence between the methods, in which the single-model cycle time c corresponds to the shift duration and the ith elemental task duration corresponds to the ith task group duration. We may achieve balances by allocating in the one instance elemental tasks to intervalsequal to the cycle time and in the other by allocating task group durations to intervals equal to the shift duration.

However, because T is the shift duration it is not possible to make changes in T in a way corresponding to changes made in c in single-model balancing. Consider a

mixed-model assembly with λ stations and a production requirement P, where

$$P = \sum_{m=1}^{p} P_m; \quad p \text{ is the number of models}$$

and where P_j is the number of units required of model j. If it is found that the task group durations A_i are such they cannot be fitted into the λ stations then either P must be reduced or λ must be increased. The method chosen will depend on whether it is more important to meet some given requirement or to avoid an increase in the number of stations.

It has already been shown that allocation of complete task groups results in allocation to a given station of every representation of a given task. However by accepting the condition that representative tasks might be allocated to one of two given stations, balance efficiency can be increased.

Consider a task group A_i that cannot be allocated to station h because the duration of A_i is too great by some amount ε work units. Let the duration of the elemental tasks that form A_i be t_i. Now, if some number n tasks is removed from A_i where $nt_i \geq \varepsilon$, then the modified task group A_i will fit into station h, and the residue nt_i

may be allocated to station h + 1. If the largest duration of any elemental task were α, then the efficiency in each station could never be less than

$$(T - \alpha)/T$$

a remarkably high value since α would seldom exceed $^1/_{200}$ of T.

Two penalties would be involved. First, the operators in station h and in station h + 1 would both need to be trained in any task that might straddle both stations. Second, there could be confusion over execution of tasks. For example consider that 7 units of model j are needed and that in the allocation of the ith task group 4 units of model j have been given to one station and 3 to another. The operator in the first station might then be required to remember that for the first four units of model j he would execute task i, and for the final three units task i would be executed in the next station downstream. A compromise would be to permit nothing smaller than the duration associated with all the units of a given model to be separated for allocation to another station. Whether or not such penalties could be acceptable would of course be a decision to be made by the industrial staff responsible for the assembly line. It is important, however, to draw the

attention of those concerned to the existence of such alternatives. To illustrate its feasibility, this method of splitting is used in the balance runs in Chapter 5.

3.3.2 Mixed-model balancing by multiple individual balances

A method, called method 1, that relies on repetitions of individual balances has been formulated in Chapter 2. This method is described in some detail [Macaskill, 12]. Comment will be limited here to a general discussion of the method and its difficulties.

Consider a mixed-model requirement P where, as in 2.3.1 above,

$$P = \sum_{m=1}^{p} P_m$$

Task durations are known and therefore the work content w_j of model m can be determined. If the efficiency of the balance for model m were known, the effort E_m that would be required for assembly of P_m units could then be determined from knowledge of the work content w_m. Finally the effort required for the whole shift, E, would be obtainable where

$$E = \sum_{m=1}^{p} E_m$$

From knowledge of E and the E_j a proportion of the shift T_j could be allocated to each set of models j where

$$T_m = TE_m/E$$

The problem would then be to balance each model by a single-model balance for production of P_m units in a shift period T_m, with the overall condition that the number of stations should be the same for each model.

Provided that balance efficiencies can be reasonably accurately predicted the following approach can be used. Determine all the T_m by the method given above, and let k be the model for which the corresponding value of T_m is greatest. Determine $c_{k\ max}$ where

$$c_{k\ max} = \frac{T_k}{P_k}$$

Carry out a single-model balance for model k with $c_{k\ max}$ as model cycle time, and let the number of stations required be λ. Then iteratively reduce $c_{k\ max}$ without increasing λ. If the reduction in c is δc, this will correspond to some amount δT that is no longer required in the partial shift duration T_m allocated to the units of model m.

Now select the model for which T_m is next greatest after model T_k. Let this be model g and attempt to balance this model to λ stations to a cycle time c_g where

$$c_g = \frac{T_g + \delta T}{P_g}$$

If the balance is achieved, reduce c_g as far as possible without causing a reduction in λ, and determine a new value for δT.

If the balance cannot at once be achieved increase cycle time until the balance can be achieved. This will entail introduction of a negative value of δT. In either case determine the model that has the next greatest shift duration and repeat the procedure again.

Continue in this manner until all models have been balanced or until the whole shift duration has been used. In the latter instance increase number of stations by 1 and start again.

If the balance is achieved and there is a sufficiently large residue of shift duration, reduce the number of stations by 1 and start again.

As explained in Chapter 2 the method has a weakness when compared with the task group allocation method in that every repetition of a given task will not usually be allocated to one particular station.

There are also detailed difficulties caused by variations of balance efficiency with cycle time. These and other aspects of the technique are discussed in the section of this Chapter relating to results from the

computer program.

3.4 Heuristic procedures

An optimal balance for a given product assembly is such that no other feasible allocation to stations of the elemental tasks of that product will reduce the operator idle time. Such balances in theory are attainable by listing every feasible arrangement of elemental tasks. This method fails in practice through the magnitude of the computation required. Dynamic programming methods have been devised [Held et al., 3 and Jackson, 7] that greatly reduce the number of feasible sequences that need be considered in obtaining an optimal balance. With these methods computational effort is prohibitive for practical assembly. Held et al. [ibid.] have also devised a dynamic program that obtains near-optimal balances with reduced requirements for computation.

Nevertheless, an attractive alternative that in many instances is faster and more practicable than dynamic programs is the heuristic procedure. The principles of such procedures are now discussed.

In any precedence graph there is at least one task that has no predecessor. Such tasks are here called free tasks and in heuristic procedures are grouped together in a set here called the active list. In general terms, the

method of the heuristic procedure is to select a task from the active list by means of a predefined procedure and to attempt to allocate it to a station. If it will fit then it is allocated and removed from the precedence graph and the active list. Tasks that become free tasks because of the removal from the precedence graph of the allocated task are then placed in the active list. If a task selected for allocation will not fit, another task is selected from the active list for a further attempt at allocation. If no free task will fit into the current station, the next station is used for allocations. By suitable repetitions of this process the situation is finally reached when all the tasks have been allocated or when the balance has failed and must be repeated with new initial conditions. A detailed explanation of the method is given in the flow chart of figure 2.

Heuristic procedures form the basis of all the assembly-line balance operations reported in this thesis and a key feature of these procedures is the nature of the rule used to select free tasks from the active list. As examples of the types of rule that can be used, two well known predefined selection rules are now described. Let F be the set of tasks in the active list. The rule is represented as the result of an operation H_q on F, where the

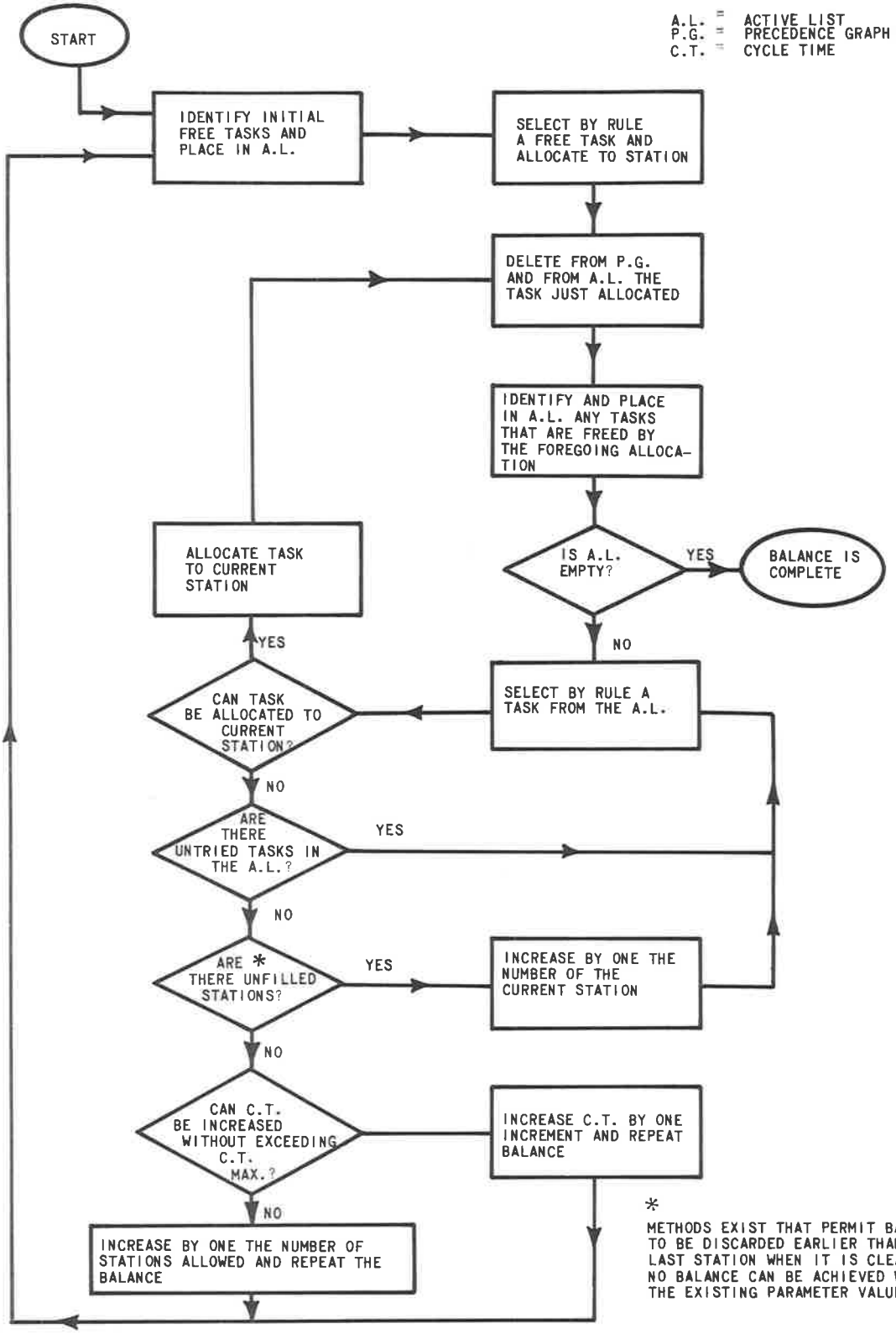


FIGURE 2 : FLOW DIAGRAM OF A HEURISTIC PROCEDURE

subscript q identifies the particular rule being used.

3.4.1 Largest candidate procedure

If H_1 is the operator for the largest task procedure [Moodie and Young, 15], we have

$$H_1(F) = \max_i \{t_i, i \varepsilon F | t_i \leq r\}$$

where r is the current value of the residue of effort available in the station to which allocations are being made and t_i is the task duration of the ith member of F. The procedure requires only that task durations should be known, and therefore requires no preliminary computation.

3.4.2 Ranked positional weight procedure

If H_2 is the operator for the ranked positional weight procedure [Helgeson and Birnie, 4] and [Mansoor, 13] we have

$$H_2(F) = \max_i \{(pw)_i, i \varepsilon F | t_i \leq r\}$$

where r is the current value of station residue of effort, where $(pw)_i$ is the positional weight of the ith member of F, and where

$$(pw)_i = t_i + \sum_{j=1}^{k} t_j b_j ;$$ k is the number of tasks in the precedence graph

and $b_j = \begin{cases} 1, & \text{if } j \varepsilon \text{ the set of successors of } i \\ 0, & \text{otherwise.} \end{cases}$

A special computation is required to determine $(pw)_i$

for all tasks in the precedence graph. Note, however, that once the $(pw)_i$ have been determined for a particular precedence graph and set of durations, the $(pw)_i$ are available without further computation for the many balances required.

3.5 Design of computer balance program

The main factors considered in the design of the computer program for the assembly line balancing were as follows:-

(i) It was clear that both during the research and in industrial use there would be a need in a mixed-model balance program for the storage of significant quantities of precedence and other data for several models.

(ii) It was also clear that both during the research and in industrial use there would be a need for frequent computer runs of the balance program with many of the runs involving several balance repetitions.

(iii) It was considered that during the research stage there would be a need for frequent changes in the computer program and therefore simplicity and flexibility would be of importance. This

need was emphasised by the need for a wide range of functions and alternatives in the program which would tend to make the program large and intricate.

(iv) Although the main aim of this research was concerned with the understanding of assembly line behaviour, it was considered important to investigate programming methods that might facilitate later industrial work.

The need for computational speed (factor (ii) above) suggested strongly that all quantities needed for the balances should be available in core store. This requirement allied with factor (i) above caused much emphasis to be placed on methods of reducing the amount of core store required. The language chosen for the program was Control Data FORTRAN. It was considered that any computational speed that might be lost through using FORTRAN instead of an assembly language would be more than regained by the ability to make changes easily and check them out rapidly. Use of a higher level language such as SIMSCRIPT was rejected partly because it was considered that FORTRAN would give more flexibility and partly because on the University of Adelaide computer the FORTRAN compiler was thoroughly checked out, whereas the SIMSCRIPT compiler and language

had been little used.

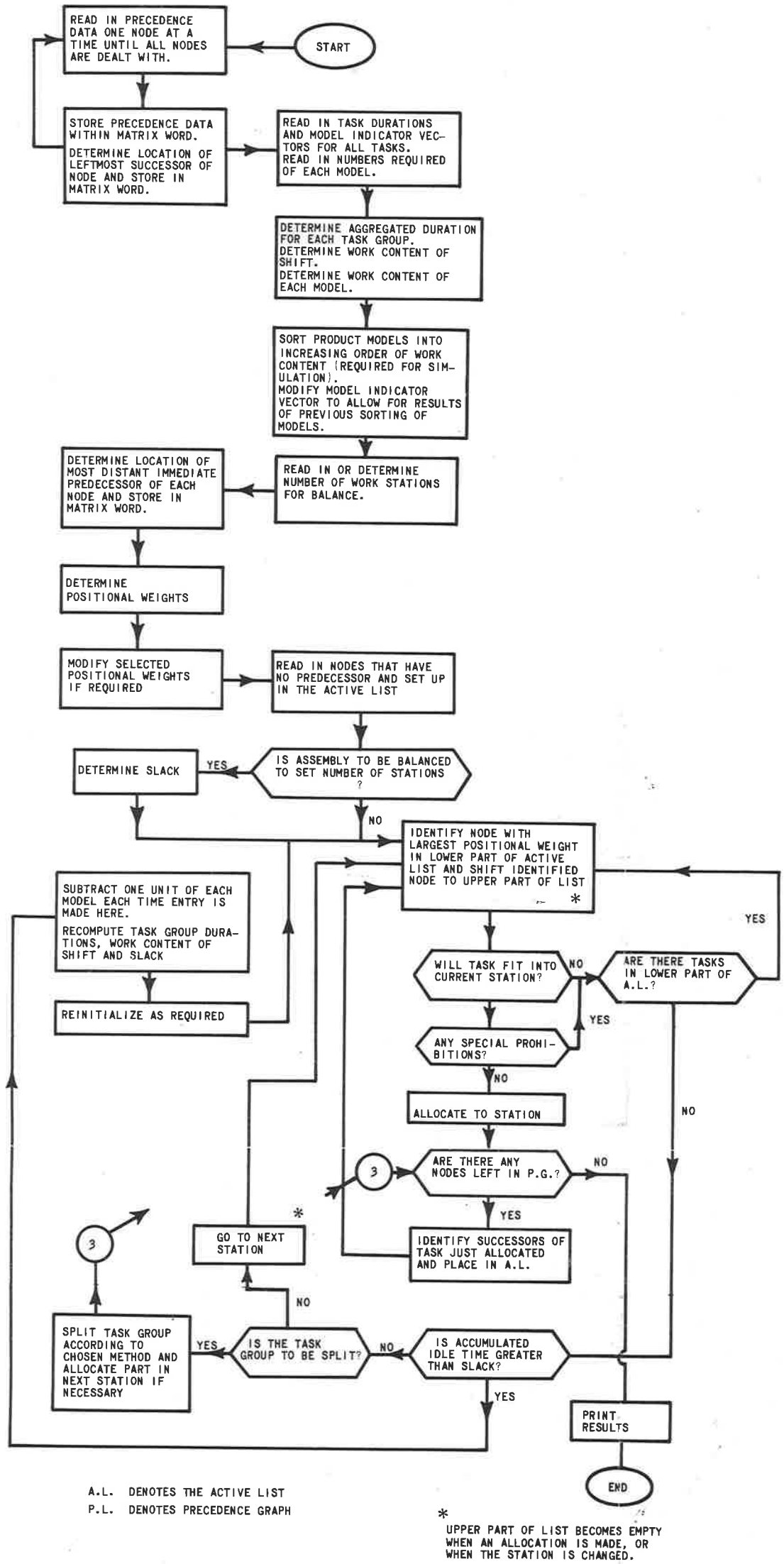
Those aspects of the balance computer program that are considered to be of most significance are now discussed. Figure 3 shows a flow diagram that illustrates the main functions of the program arranged for balancing by aggregated task groups by use of the ranked positional weight (r.p.w.) procedure.

3.6 Identification of free tasks

A method for identifying free tasks by Boolean operations in a precedence matrix has already been described [Macaskill, 12] and is included in the balance program. This method although simple in concept and satisfactory for small precedence graphs is not well suited to use with large assembly lines. The method now described is a development of the foregoing method and is faster, more economical and well suited to use with large assembly lines. It is the method that has been used in the solution of the practical problem reported in Chapter 5 of this thesis.

3.6.1 Modified precedence matrix

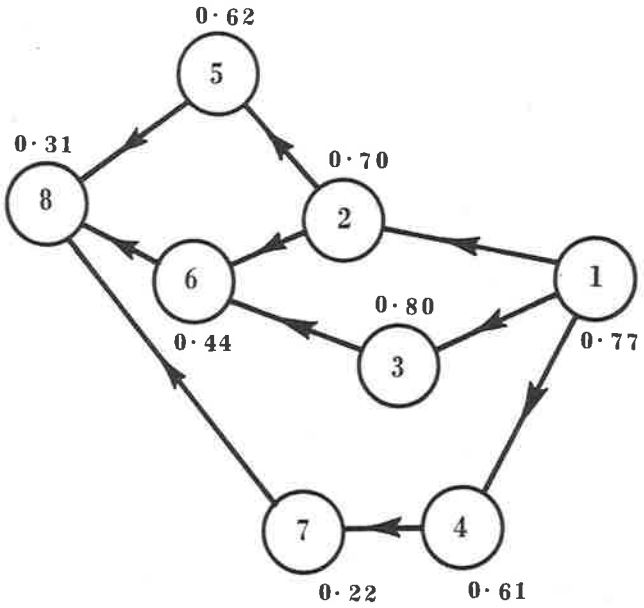
Consider the precedence graph shown in figure 4a. The precedence information of the graph is contained in the precedence matrix, A, of figure 4b. Row i of the matrix corresponds to node i of the graph and contains the



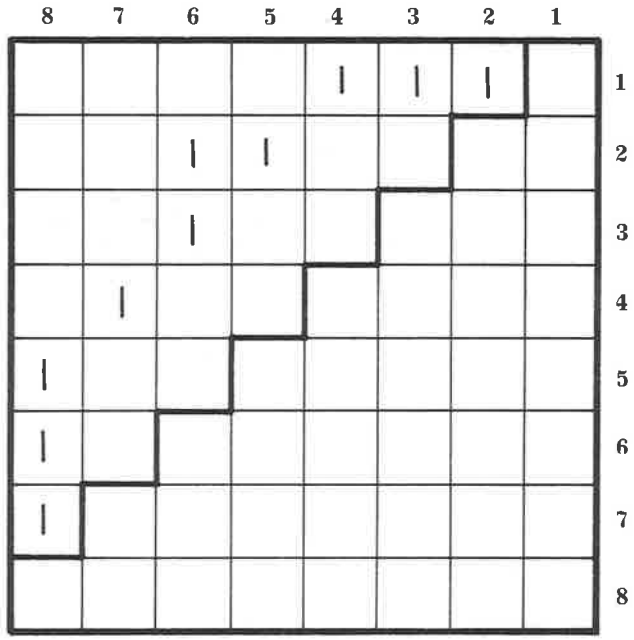
A.L. DENOTES THE ACTIVE LIST
P.L. DENOTES PRECEDENCE GRAPH

* UPPER PART OF LIST BECOMES EMPTY WHEN AN ALLOCATION IS MADE, OR WHEN THE STATION IS CHANGED.

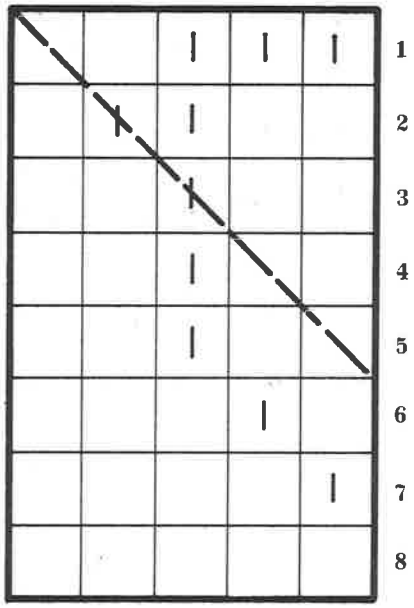
FIGURE 3 : GENERAL FLOW DIAGRAM FOR ASSEMBLY LINE BALANCING PROGRAM.



a. PRECEDENCE GRAPH



b. PRECEDENCE MATRIX



c. MODIFIED PRECEDENCE MATRIX

FIGURE 4 : METHOD OF STORING PRECEDENCE DATA

immediate successors of node i in locations corresponding to their node numbers. Let a_{ij} be the element of A that lies in row i and column j (column 1 is the rightmost column of A). Form a new matrix B such that its elements b_{ij} that appear in row i and column j are the elements $a_{i(j-i)}$ in matrix A. Figure 4c shows the matrix B formed from the matrix A of figure 4b. In matrix A the immediate predecessors of node j lie in column j. In matrix B the immediate predecessors of node j lie in the locations $b_{j-1,1}$, $b_{j-2,2}$, and so on along the diagonal. The diagonal relating to node 6 is identified in figure 4c.

In a heuristic procedure, when a node i has been allocated to a station, all its immediate successors that have no remaining predecessor may be entered in the active list. The following procedure identifies such nodes

(i) identify all the immediate successors of the allocated node by searching for entries in row i of B. If the location of the leftmost successor is known, the amount of searching can be reduced significantly,

(ii) delete row i of B,

(iii) search in turn along the predecessor diagonal of each successor of i. It is helpful if the

location of the entry furthest along the diagonal is known, for the search need never be carried beyond this location. If an entry is found before reaching this location the search is terminated; if no entry is found then the relevant node is placed in the active list.

3.6.2 Computer storage of precedence information

The foregoing method is particularly valuable for a computer application for it facilitates economic storage of large precedence graphs. In the computer program used in Chapter 5 one computer word (each word has 60 bit positions) is allocated to each node. The successors of the node are recorded in the first 30 bit locations of the computer word. (If node (i+a) immediately follows node i where a > 30 a dummy node with task duration zero is inserted in the precedence graph.) In addition the locations of the leftmost successor and the predecessor furthest along the diagonal are stored in the node word. The procedure noted in (i), (ii) and (iii) of the preceding section is then applied in the computer program by use of FORTRAN masking expressions. The method used is illustrated in the fully annotated flow diagram of figure 5.

The bit locations allocated to successors have been

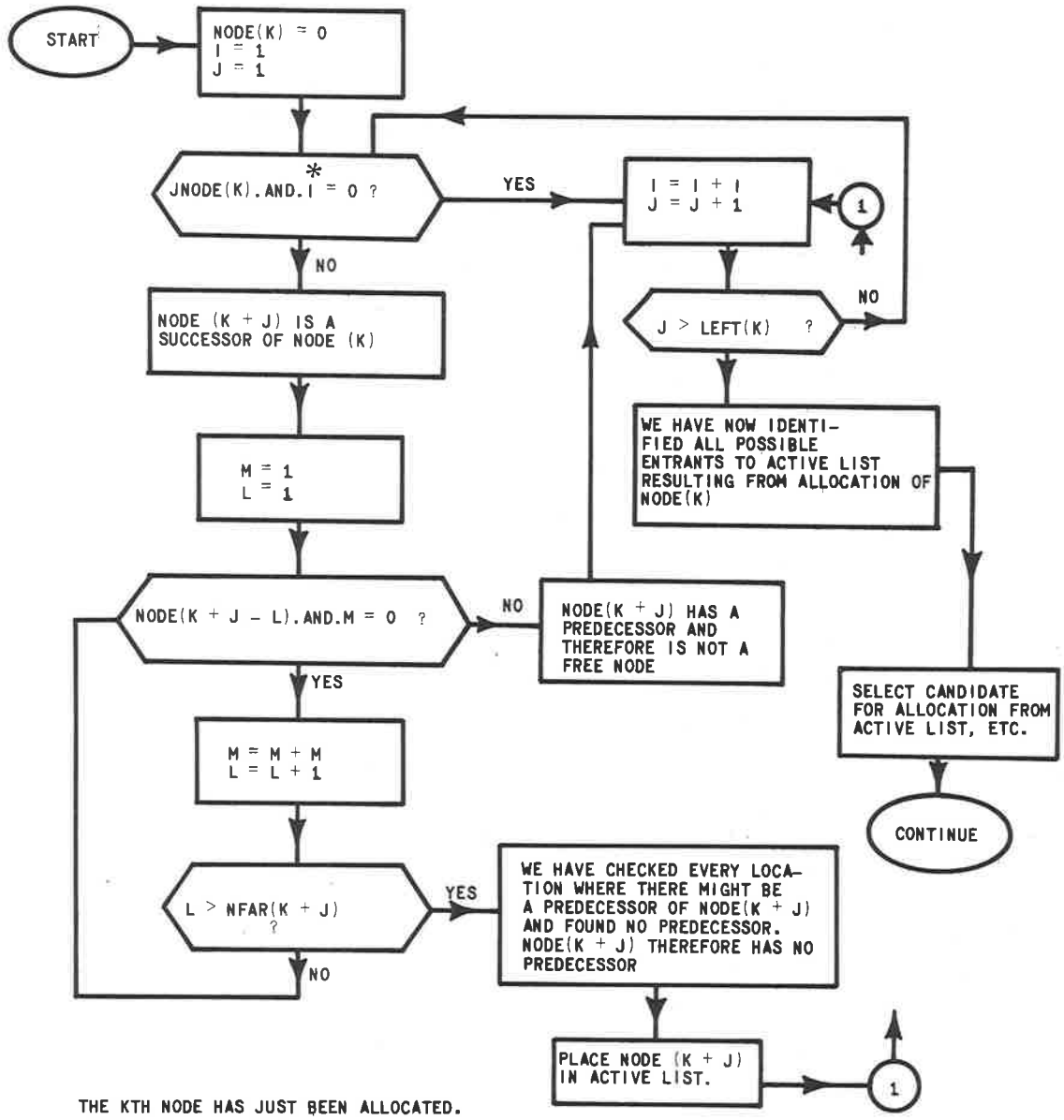


FIGURE 5 : METHOD OF IDENTIFYING FREE NODES IN PRECEDENCE GRAPH

limited to 30 to show that the method is practicable without the use of all locations in a very large computer word. Because the precedence matrix B is destroyed (see item (ii) of the procedure) an "active" and a "passive" version of B are held. If repetitions of the balance are required the passive matrix is used to recreate the active matrix.

In the example given in Chapter 5 a balance (one iterative pass only) is achieved for a 190 node precedence graph in less than 0.5 sec. This computational speed is considered adequate.

3.7 Determination of positional weights

Positional weights have already been defined (3.4.2). As examples of such weights consider figure 4a. We have

$$(pw)_3 = t_3 + t_6 + t_8$$
$$(pw)_2 = t_2 + t_5 + t_6 + t_8$$
$$(pw)_4 = t_4 + t_7 + t_8$$

The set of all the successors of a given node n_i in a graph is obtainable as follows. Determine the set of immediate successors of n_i the set N_1 say. Then determine the set of successors for each member of N_1, the sets N_2, N_3,...N_j. Then determine the sets of successors of every member of each of the sets N_2, N_3,...N_j. Continue

in this manner until no further successors are to be found.

The application in the programs of this method is given in the flow diagram shown in figure 6. This diagram contains sufficient explanatory matter for it to be applied to an example (e.g. figure 4a) if required. The computational speed of this algorithm is illustrated by the computation made for the positional weights of the 191 node precedence graph used in Chapter 5. The central processor usage for this computation was about 1 second which is considered acceptable.

3.8 Computations using slack

In Section 2 of this Chapter iterative single-model balances in which cycle time is increased step by step until a balance is achieved are discussed. In Section 3.3.1 a similar iterative approach is used in which mixed-model balances are achieved by iterative reduction of the production requirement. A method is given [Mansoor, 13], for single-model balances, for the abandonment of a balance iteration as soon as it shows itself to be unsuitable. This method has been applied directly in the program for single-model balances and an adaptation of the method is included in the mixed-model balance program as follows:-

Let T be shift duration and λ the number of stations that may be used, then

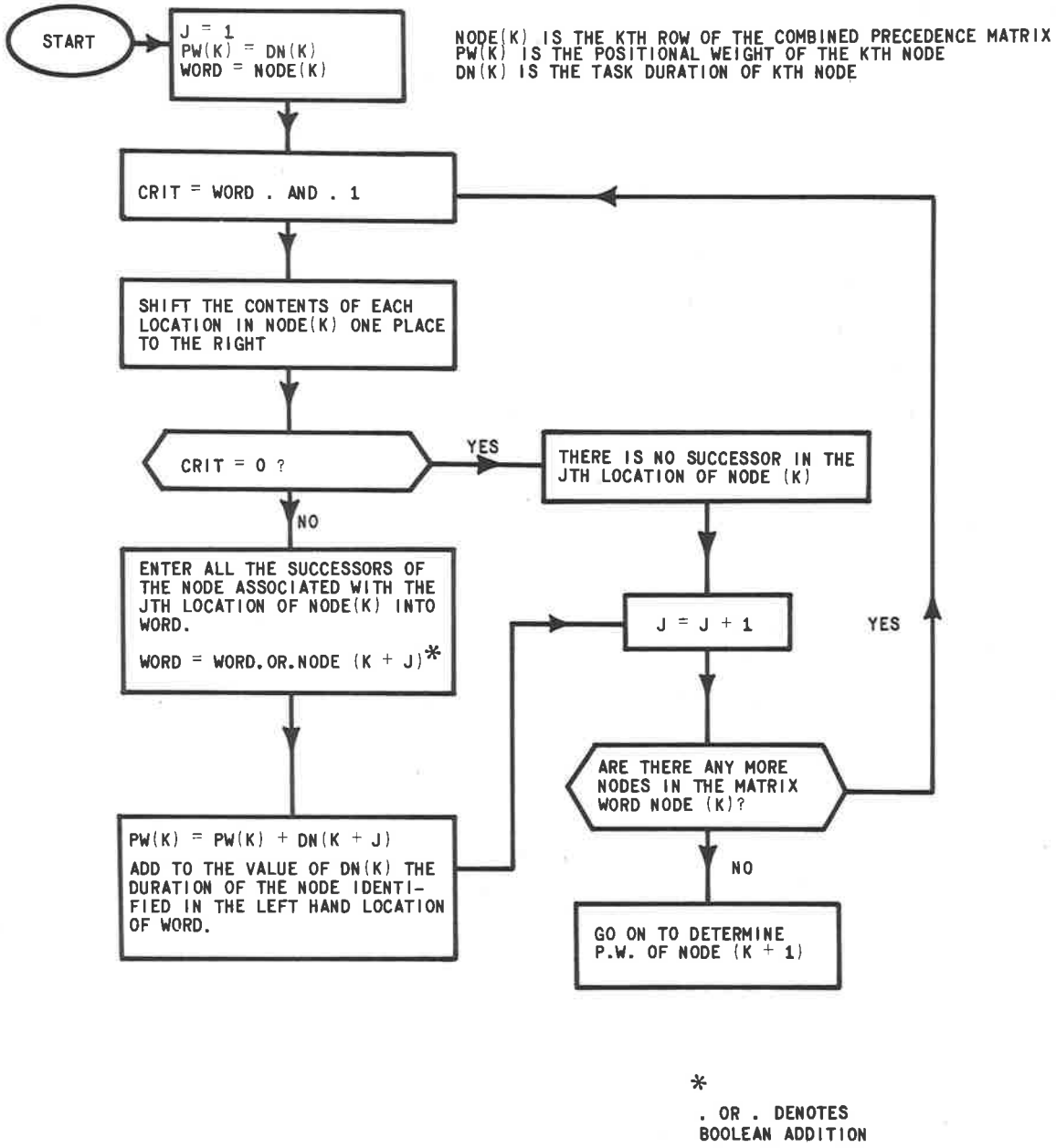


FIGURE 6 : FLOW DIAGRAM TO ILLUSTRATE COMPUTATION OF POSITIONAL WEIGHT OF THE KTH NODE IN THE PRECEDENCE GRAPH.

Assembly effort supplied = λT

Work content of the shift = $\sum_{i=1}^{k} A_i$, k is number of tasks.

The difference between effort supplied and the shift work content is called the total slack and is denoted by R. We have

$$R = \lambda T - \sum_{i=1}^{k} A_i$$

Now let τ_j be the aggregate duration of task groups allocated to station j. Then the slack associated with station j is $T - \tau_j$ and the slack associated with the complete allocations to the first h stations is

$$\sum_{i=1}^{h} (T - \tau_i)$$

It is clear that if the slack associated with some given number of stations exceeds the total slack, then more effort will be required than has been supplied and a balance will never be achieved. That is, if

$$R - \sum_{i=1}^{h} (T - \tau_i) < 0$$

the attempt to balance may be discontinued. The production requirement is then decreased by 1 chosen product unit and another balance iteration is started.

3.9 Results of balance program runs

In the research reported in this thesis it has been considered more important for the balance program to produce balances of acceptable efficiency quickly and effectively than to produce balances of very high efficiency. Consideration of the following matters contributed to this decision.

(i) It was judged that overall performance of a mixed-product assembly line would be much more sensitive to product sequence fed to the line than to balance quality.

(ii) The aggregated task group method of balance was thought likely to lead to a very high balance efficiency with quite crude heuristics provided that task group splitting was permitted. Even without this splitting it seemed probable that the balance efficiency would be well within acceptable limits because the number of free tasks associated with the combined precedence graph would be much greater than for a single-model.

(iii) In the multiple individual balance method it was considered that the distribution of station idle

time would differ from model to model. As a result efficiency for all the balances would probably be significantly higher than for any one of the individual single-model balances considered separately.

(iv) Powerful techniques are available [Arcus, 1],[Hel Karp and Shareshian, 3] and [Mansoor, 13] for achieving high quality balances if required. It was thought however that the presence of these more complicated techniques when combined with some of the special functions of the program would greatly complicate the balance procedures without significantly adding to the research results.

Despite the foregoing comments several single-model balance runs were made with the object of gaining an understanding of any performance anomalies that might interact unfavourably with the application of single-model balancing concepts to mixed-model balancing techniques.

Some of the findings ofthis work are of general interest and are now reported.

3.10 Balance-simulation experiment

There are two main aspects which affect the performance

of a simple heuristic procedure, both of which depend on the task selection rule of the heuristic. First, the way in which tasks are chosen from the active list determines the nodes that are released from the network and hence determines the subsequent content of the active list. Second, the durations of the tasks chosen will determine how closely the sum of the tasks allocated to a station will approximate to the cycle time.

Examination of the results of running of the balance program suggested that a heuristic procedure that gave good results in relation to the first aspect described above might give poor results in the second and vice versa. For example, the very procedures that established an active list with a large membership might result also in a less useful selection from that list than the selection made from the active list by a procedure unable to establish so long an active list. It was decided to investigate this matter and after some deliberation simulation was selected as the most practicable method of investigation. The experiment and its results are described below for a comparison of the largest candidate heuristic with the positional weight heuristic.

The basis of the method was to set up an active list that throughout a simulated assembly-line balance would

contain some fixed number of members. With this as a basis there would be no need to consider any specific precedence relationships, for the general nature of the precedence relations of the balance would be implicit in the maintenance of a constant number of members in the active list.

The distribution of the durations of the members of the active list reflects the nature of the assembly operation being undertaken. In the simulation the durations were selected randomly from a rectangularly distributed population of numbers lying between 0.01 and 0.99 time units.

3.10.1 Simulation of largest candidate heuristic

In these conditions the simulation of the largest candidate heuristic is simple, as follows.

(i) A cycle time is defined. Ten different cycle times were used with starting value 1.10 time units and incremented by 0.10 time units to a final value of 2.00 time units.

(ii) The length of the active list is defined. In this experiment the list length could have the values 2, 4 or 6.

(iii) An active list is set up by selecting six rectangularly distributed random numbers lying

in the range 0.01 to 0.99.

(iv) The largest of these numbers is then selected and allocated to the station, and station residue of effort is determined by subtraction of this number from the chosen value of cycle time. The active list member is deleted and another random number is generated and added to the list.

(v) Again, the largest active list member is selected and then tested for fit in the station residue. If it will not fit the second largest is tried, and if that will not fit, the next largest until a fit is obtained or it has been established that all active list members have durations greater than the station residue. In the latter instance, it is necessary to move to the next station - achieved in practice by resetting the station residue to a value equal to the cycle time.

(vi) The procedure described above is repeated until 40 nodes have been selected. Subterminal efficiency (3.2.2) is then computed for the balance.

Each procedure in 1 to 6 above was repeated 50 times for each of ten cycles times for each of 3 different lengths of active list making a total of 50 x 10 x 3 = 1500 runs.

3.10.2 Simulation of ranked positional weight heuristic

The simulation of the ranked positional weight heuristic was done as follows. Consider a six member active test F = {a,b,c,d,e,f} ranked in order of positional weight. From this test, a will be chosen first for allocation, and if its duration is too great b will next be tried, then c and so on. If a were allocated, and the list were not augmented, the next set of allocation attempts would be made in the order b, c, d, e, f, because this is the order of the positional weights. However, the requirement of the simulation is that the active list shall have a constant number of members, and the question is how to locate the new member in the list. Examination of accession of tasks to the active list in some practical networks showed that a new member might take any place in the positional weight ranking with the one exception that it was not usual for the new member to have a positional weight greater than the member at the head of the list. The following procedure was therefore selected as a sufficiently accurate simulation of the positional weight heuristic for the purposes of the experiment.

(i) The initial order in F was taken as the order of generation of the first set of members of F.

(ii) Each later addition to F was given a duration by generation of random numbers as for the largest candidate procedure. The position in F of the additional member was defined by selection from a population of random numbers rectangularly distributed over the range (2,3,...,n) where n was the number of members of F. For example if n=6 and the random number chosen was 4, then the new member would be placed in position 4 in F and the members in positions 4 and 5 after allocation would be pushed down to positions 5 and 6.

Once F had been established selections were made from it by trying each member for allocations in the order of their location in the list, either until an allocation could be made or until it was established that none would fit. From this point the procedure was similar to that for the largest candidate simulation, 1500 runs being made in the same conditions.

Finally, an investigation was carried out for a heuristic $H_3(F)$, called here the smallest candidate heuristic

where

$$H_3(F) = \min_i \{t_i, i \epsilon F\}$$

This investigation was limited to a 6 member set F.

3.11 Results of balance simulation

In figure 7 average subterminal balance efficiencies obtained in the simulation are plotted against cycle time for the various conditions investigated. The results are of considerable significance as discussed below.

(i) For all three heuristic procedures the average efficiency showed a significant and regular increase with cycle time and (for the two procedures where the comment is relevant) with membership of active list.

(ii) In all instances the largest candidate procedure showed an advantage over the positional weight procedure. This is consistent with the hypothesis that provided that the active list lengths are the same, the largest candidate method of selection is the better. This is supported by the exceedingly poor results obtained with the smallest candidate procedure and a six-member active list.

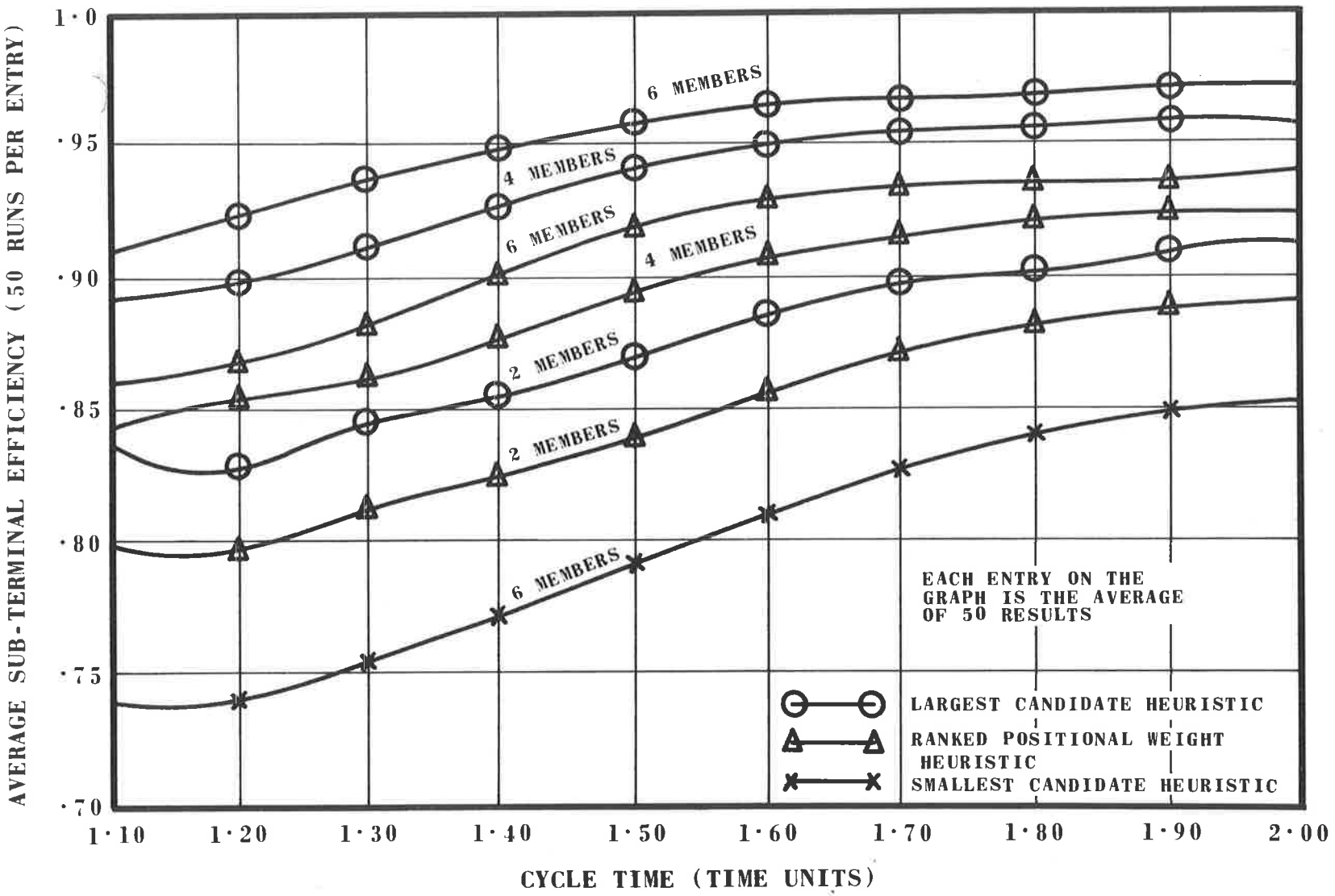


FIGURE 7 : RESULTS OF BALANCE SIMULATION EXPERIMENT

(iii) Unfavourable balances will be obtained in the average with a cycle time that is not greater than 1.5 $\max_i(t_i)$ where the t_i are the task durations in the network and have an approximately rectangular distribution.

The variance of the balance efficiency was computed for each set of 50 runs. The maximum variance recorded was 0.002 at c = 1.4. The distribution of balance efficiency is a skew distribution, and it is estimated roughly that for this value of variance, 1 balance efficiency in 40 would be 10% or more less efficient than the average efficiency for the set of 50 runs. Starting from the value of cycle time c = 1.4, variance decreases with increasing cycle time and also decreases as the active list membership is increased. At c = 2.0 the maximum variance for 4 and 6 member lists was slightly greater than 0.0005.

The significance of these results concerning variance is that with heuristic balances, especially in unfavourable conditions, efficiency of balances obtained from simple heuristic procedures will tend to show a wide variation in efficiency. This variation in efficiency corresponds to the wide range of quality of balance obtained by Arcus [1] in his randomly generated sets of balances.

3.12 Test balances

For reasons given in Section 3.9 of this Chapter no attempt was made to study heuristic procedures exhaustively. However some running was done for some test assembly operations and an example of one of the precedence graphs used in this work is shown in figure 8. The average number of free tasks obtained with different values of cycle time for this test product is shown in figure 9, and the values of balance efficiency for the same product are shown in figure 10. The heuristics used were:-

(a) Largest candidate heuristic.

(b) Ranked positional weight heuristic.

(c) A heuristic in which the criterion for ranking was the total duration of the task and its immediate successors. This was called the first successors heuristic.

(d) A heuristic in which the method of the positional weight heuristic was followed in each station until the first occasion on which a task would not fit. For the remainder of the station the largest candidate procedure was then used. This was called the mixed heuristic.

(e) A heuristic that differed from the mixed heuristic only in that the first task allocated

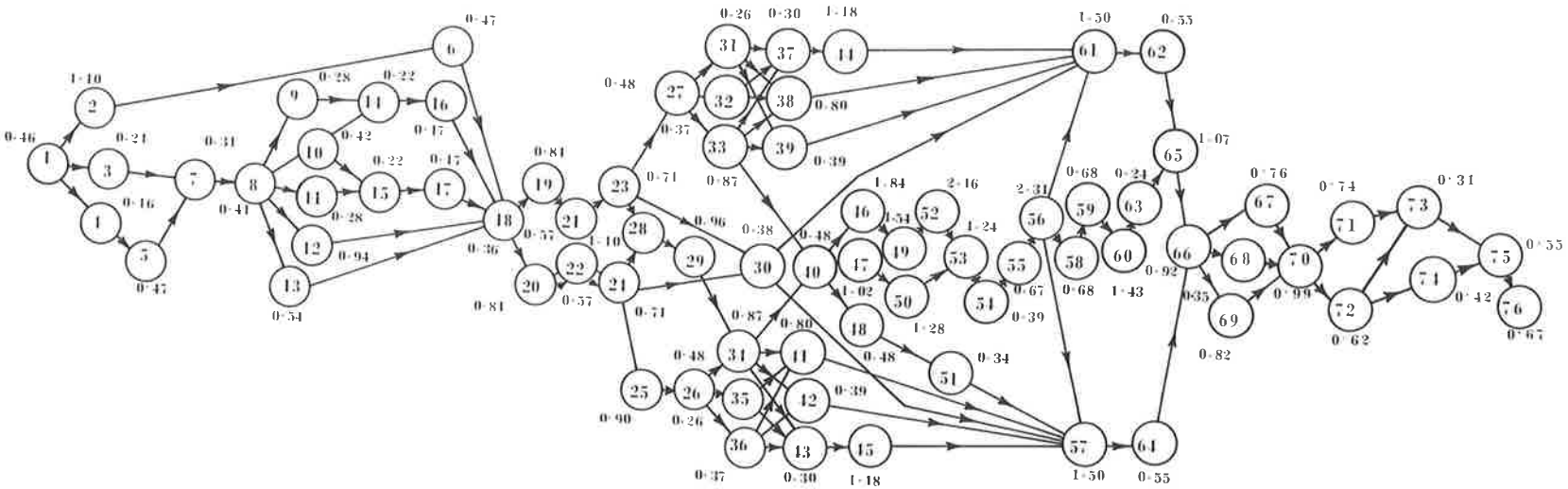


FIGURE 8 : PRECEDENCE GRAPH FOR TEST PRODUCT 1

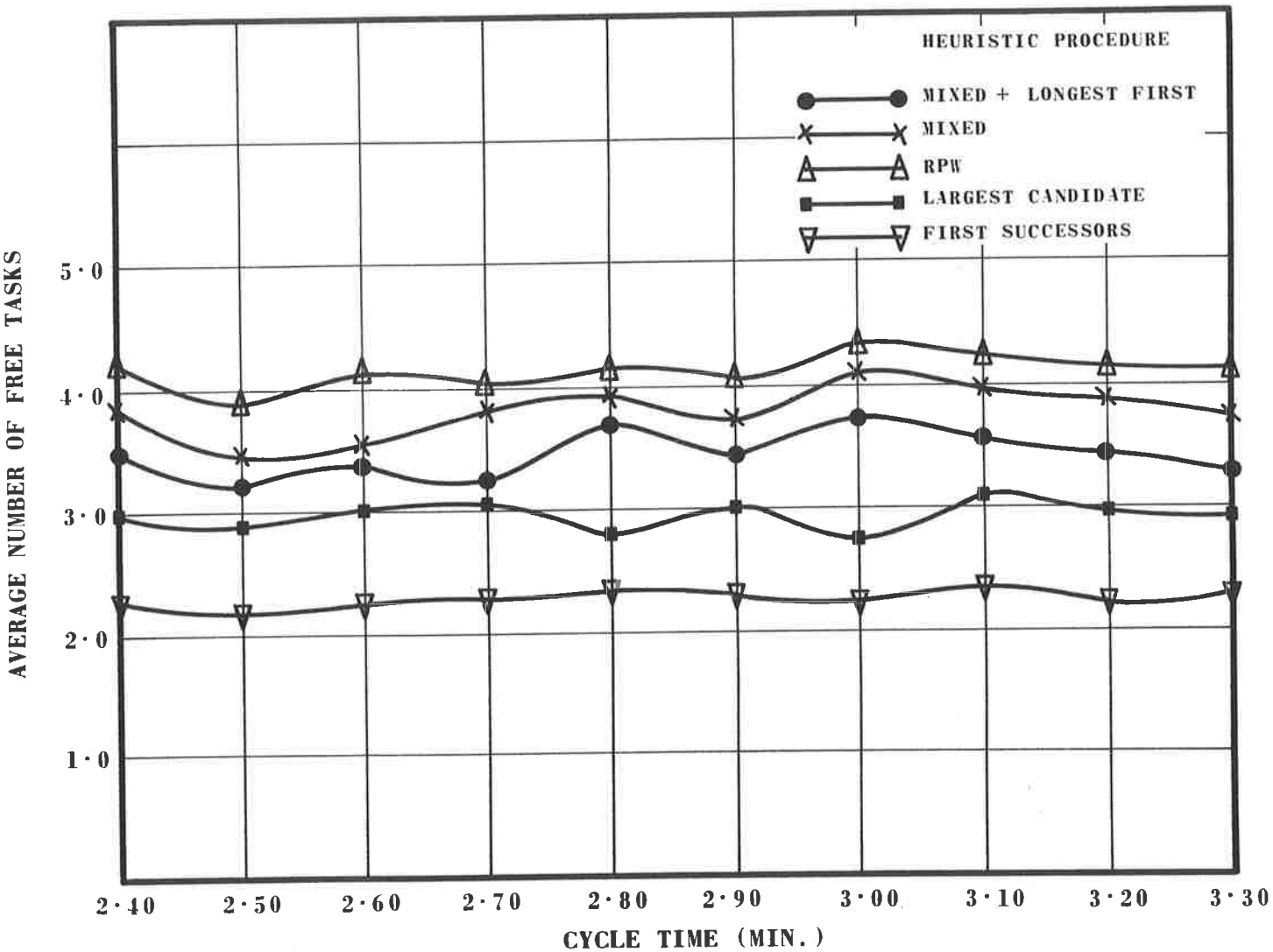


FIGURE 9 : AVERAGE NUMBER OF FREE TASKS AT ALLOCATION TIME PLOTTED AGAINST CYCLE TIME FOR TEST PRODUCT 1 FOR VARIOUS HEURISTIC PROCEDURES.

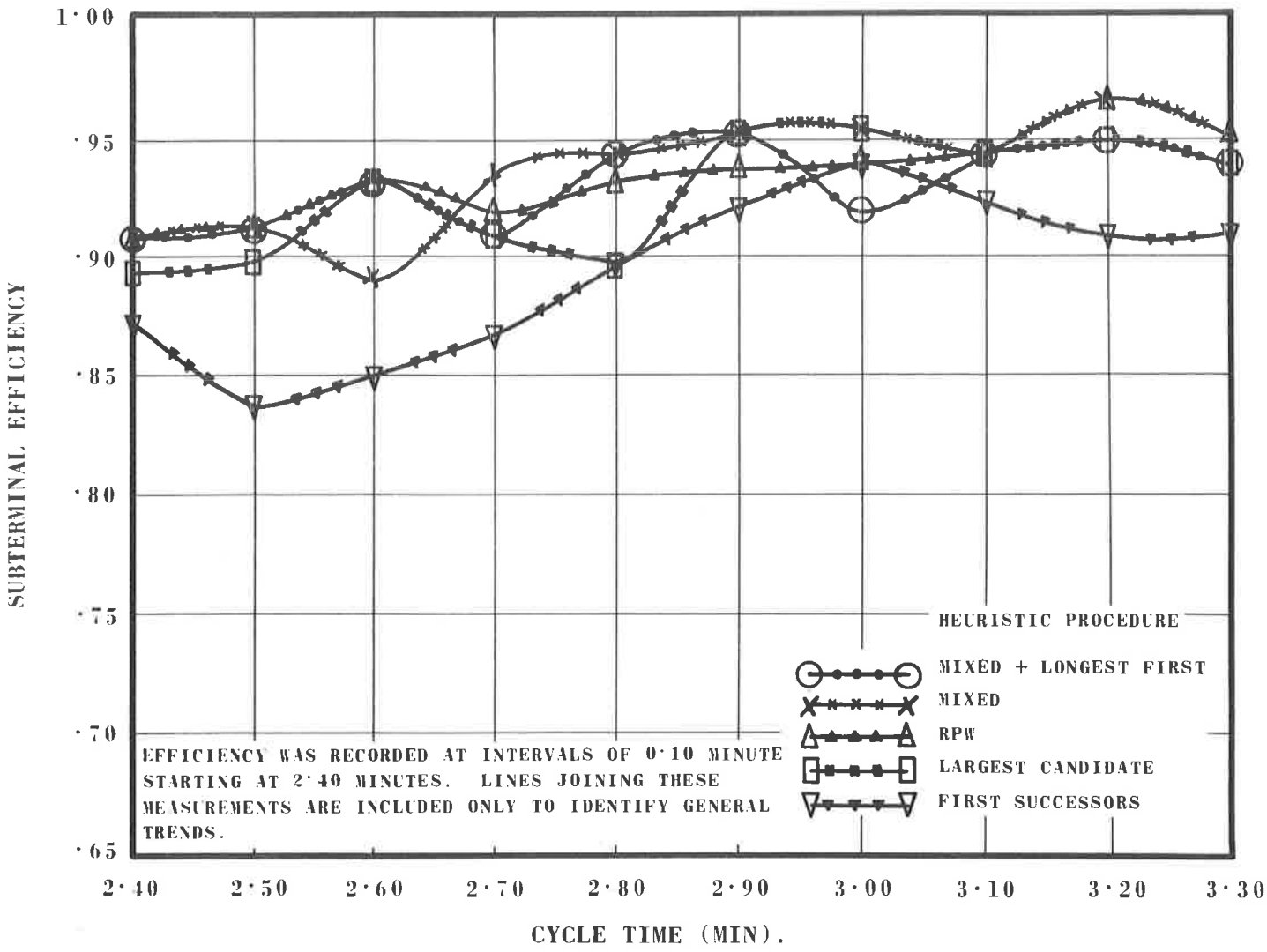


FIGURE 10 : SUBTERMINAL BALANCE EFFICIENCY PLOTTED AGAINST CYCLE TIME FOR TEST PRODUCT 1 FOR VARIOUS HEURISTIC PROCEDURES.

in each station was the task of longest duration on the active list. This was called the mixed + longest first heuristic.

The results show a general agreement with the results of the simulation. The following detailed matters are noteworthy.

The first successors heuristic has a particularly poor average length of active list and an inferior method of selecting from this list. Its efficiency record in figure 9 shows that for the instance investigated it was the worst procedure.

The largest candidate procedure shows the greatest scatter of results. This would be expected because greater changes in order of allocation would be experienced with this heuristic than with the positional weight and related procedures. Successive balances will therefore have less in common with each other and might be expected to show greater variation in efficiency.

There is little to choose between the remaining heuristics and indeed, at higher cycle times the largest candidate procedure holds its own.

The heuristic method chosen as satisfactory for the provision of the mixed-model balances required as inputs for the computer sequencing simulation was the

ranked positional weight (r.p.w.) technique. It was preferred to method 3.12(c) on account of better efficiency and to (d) and (e) on grounds of comparable efficiency and greater simplicity. For large cycle times there is little to choose between the r.p.w. and largest candidate methods and the latter is significantly simpler. Nevertheless, the ability to change the criterion of choice is often useful, particularly for special requirements. This may be done quite arbitrarily with the r.p.w. technique without affecting the parameters of the balance. In the largest candidate technique task duration and criterion of selection are defined by the same number and this precludes arbitrary alterations in the criterion.

3.13 Multiple individual balance program results

An algorithm has been successfully developed for mixed-model balances by the method discussed in Section 3.3.2 of this Chapter. One of the difficulties encountered in the method is illustrated in figure 11. Consider a 17 station balance. The range of cycle time over which the balance solution remained as a 17 station balance was about 0.02 minutes. This could prove an embarrassment if the mixed-model balance solution was 17 stations. However, in a 20 station line the loss of productivity from one station for a model of which 20 units were required with product

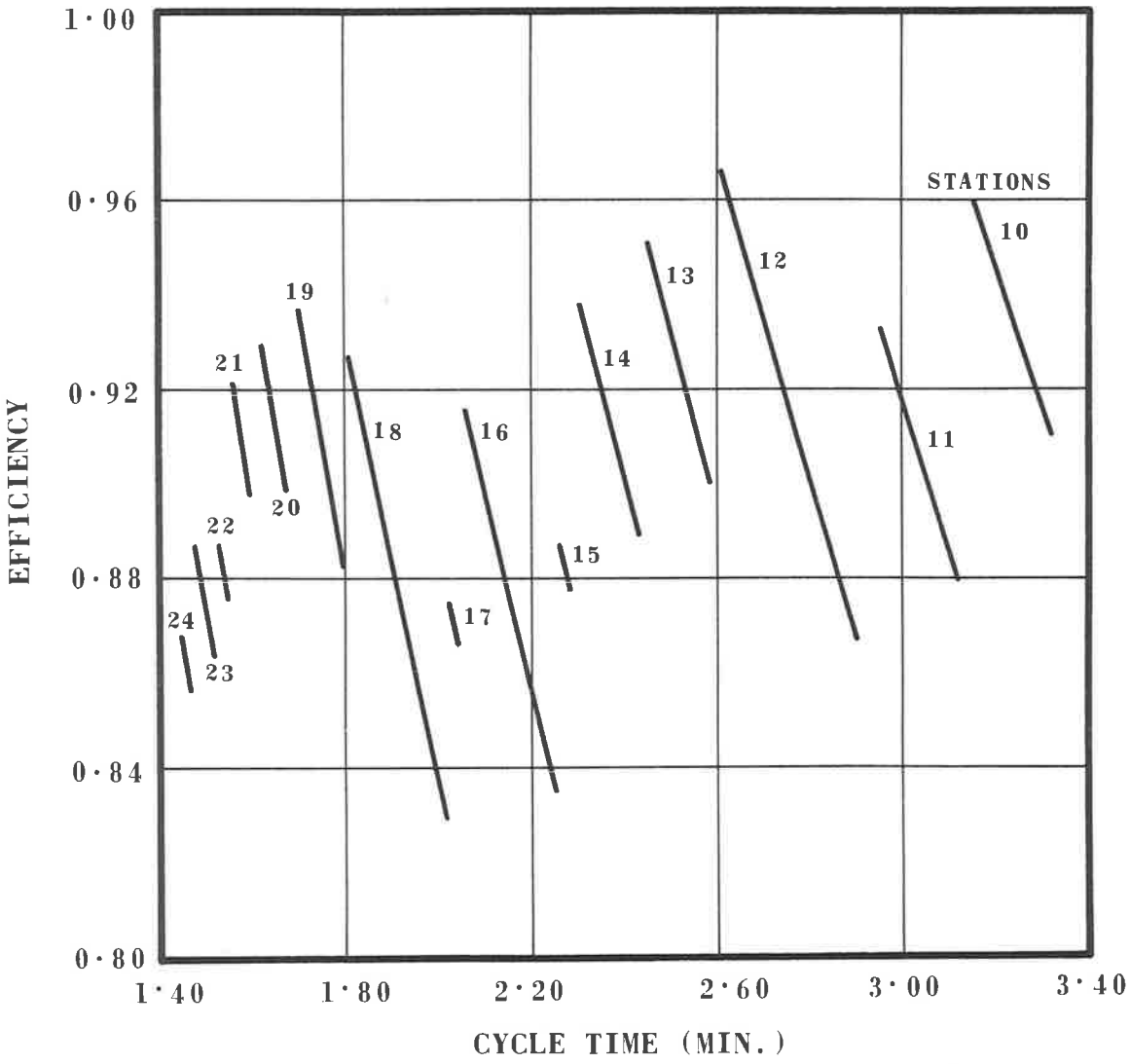


FIGURE 11 : OVERALL BALANCE EFFICIENCY OF PRODUCT 2 VERSUS CYCLE TIME

cycle time of 2 minutes would be (20 x 2/20 x 480), and this entails a drop in efficiency of less than 0.5%.

The multiple individual balance solution was not used in the practical problem solved in Chapter 5 because the nature of the models made the aggregated task group method more suitable. However, if three iterations are allowed per model, the estimated time of the balance for 15 models of 60 to 80 tasks including determination of r.p.w. would be about 15 seconds based on results achieved for three models and the latest modifications to the program. This is considered to be an acceptable speed of computation.

3.14 Special requirements

In almost any assembly line there will be special requirements. Examples are:-

(i) A particular task (or tasks) that requires to be allocated to a particular station (or stations).

(ii) A particular task (or tasks) that requires to be allocated within a particular set (or sets) of stations.

(iii) Tasks that require the simultaneous work of two operators. In order to minimize operator

idle time, it will be desirable that all such tasks be allocated to a particular station or small group of stations.

(iv) Situation in which no member of a given set of tasks may, on account of the nature of the work involved, be allocated to the same station as any member of some other given set of tasks. It will often be the case that members of some third set of tasks may be allocated to stations that contain members from either of the two incompatible sets.

All these situations are catered for in the balance program. method used is to associate attributes with tasks and check before allocation that the attribute is properly catered for if the allocation takes place. In general, this will call for every allocation to be checked for attributes, and if the attributes are of different kinds a whole array of checks may be required. This is not all however. The balance parameters may be such that the requirement specified cannot be met. For example, if cycle time is 3.00 minutes and a given task has predecessors with combined durations greater than 3.00 minutes, no balance can be achieved if the given task is to be allocated to station 1. Difficulties of this sort may

sometimes be resolved by design changes that permit suitable modifications to be made to precedences, provided that such changes are both feasible and acceptable.

In other instances, although the special requirement may be feasible, the results from the heuristic method may prevent the balance being achieved. For example, values of positional weights may be such that unless they are changed some specified station will be given its full complement of work before a task required to be allocated to it is even freed for allocation.

In this latter type of instance it will often be possible to resolve the difficulties by arbitrarily making changes in the values of positional weights. For example, if a task is being released for allocation too late for a special requirement to be met, it will often be sufficient to determine every task that is a predecessor of the task concerned and to increase the positional weight of each of these tasks. The increases should be large enough to ensure that these predecessor tasks will be chosen for allocation before other tasks that are in competition with them.

It may also be possible to achieve some special aims by introducing additional constraints within the precedence graph. As an example let a set of tasks A be such that its

members may not be placed in the same station as any member of a set of tasks C. Members of a set B may however be placed in a station with members either of A or of C. If now new arcs are included in the precedence graph in such a manner that every member of B becomes an immediate successor of every member of A and that every member of C becomes an immediate successor of every member of B, then no member of C can be allocated before any member of B and no member of B can be allocated before any member of A. If then it happens that the combined durations of the members of set B are greater than the cycle time then the requirement for separating the sets will always be met.

If the combined duration of the set B is less than the cycle time then members of A, B and C may occur in the same station and constraints will be required within the program to avoid this possibility. Even with this addition the method provides a simple solution of a problem that would otherwise introduce considerable complication into the program. Sets of type B are here called separator sets and an example of the use of a large separator set is given in Chapter 5.

In a large assembly operation there may be several special requirements to be met. If the method of solution

chosen is to attempt in the program to cater for every possible need, then the program will tend to be complicated and difficult to alter. Also, it is difficult to predict and cater in a program for the full circumstances of every special requirement and therefore unplanned program changes are likely to be required.

Consideration of this problem suggests that there will be a need for staff associated with the application of computers to assembly lines to understand the implications both as industrial engineers and as computer applications analysts. Decisions will frequently be required as to whether in a given situation it will be better to make changes in the computer program or in the physical system or in both. Such decisions are greatly facilitated if the person responsible has the training and experience to supply answers from both the engineering and the computing standpoints. Further, in such conditions the computer programs can be kept simple which in turn further facilitates the making of any changes that may be required.

CHAPTER 4. COMPUTER SIMULATION PROGRAM

4.1 Introduction

4.1.1 Definitions

It is convenient to present now certain technical definitions and explanatory material relevant to the computer simulation of the assembly line as listed below. The nomenclature, in general, follows that of Wester and Kilbridge [22] and Thomopoulos [19]. The algebraic notation for the definitions is given in appropriate places later in the Chapter.

(i) Assembly station

An assembly station is an area in which an operator is permitted to carry out his allocated tasks. In special circumstances he may work outside the station in the direction opposite to the flow of work (upstream) as far as an upstream allowance limit and in the direction of the work flow as far as a downstream allowance limit. The regions outside the station but within these allowance limits are called the upstream and downstream allowance regions.

(ii) Magnitude of station and allowance region

The speed of the assembly line is kept constant

throughout the production shift. As a result it is convenient to measure distances in terms of time taken by a product to move over them. Thus the station passage time is the time taken for a product to move through an assembly station and upstream and downstream allowance times are times taken by a product to move through the allowance regions concerned and are measures of the magnitude of the allowance regions.

(iii) Operator idle time

Operator idle time occurs when an operator has completed his task on one unit and is forced to wait before he can start work on the succeeding units. Operator idle time is related to the operator in a specific station.

(iv) Utility work

Utility work occurs when a worker reaches the downstream allowance limit of his station with his allocated tasks incomplete. He must stop work on reaching this limit and the residue of his tasks is called utility work. Completion of this work will be undertaken by a worker who is not allocated to a specific station.

(v) Congestion

Congestion occurs during the whole of the period that an operator works on a product in the downstream allowance region of his station.

(vi) Work deficiency

Work deficiency occurs during the whole of the period that an operator works on a product in the upstream allowance region of his station.

(vii) Station penalty work

The expression station penalty work is used as a collective expression relating to the quantities defined in (iii) to (vi) above.

All types of station penalty work are measured in operator-time units (e.g. man-minutes). Throughout this thesis it is assumed that each assembly station is manned by one operator unless a specific statement is made to the contrary.

4.1.2 Previous work

A study of mixed-model assembly was presented by Wester and Kilbridge [22] in 1963. In this it was assumed that work on each model could be evenly divided between the assembly operators, and fixed-interval and variable-interval launching

were examined. The fixed-interval launching was shown to be sensitive to the sequence in which models were fed to the line. By contrast, in variable-interval launching the intervals could be so arranged that any sequence of models could be accommodated without difficulty. It was noted, however, that variable-interval launching has the serious disadvantage that the attainment of the required launch intervals will entail frequent repositioning of product carriers.

In this work Wester and Kilbridge introduced the concept of congestion and idle time (defined in 4.1.1). Of these, idle time was considered to be the more harmful, and an algorithm was formulated that, in the environment of a perfect assembly balance, effectively excluded idle time and reduced congestion to a trivial level.

The work of Thomopoulos [20] was presented in 1966, and followed on logically after that of Wester and Kilbridge [ibid.]. The concepts of work deficiency and utility work (defined in 4.1.1) were introduced and a balance method (based on the same principle as that given in Chapter 2.4.5) was used, in which work on a given model was not necessarily shared evenly between stations. A mathematical model of an assembly line together with an algorithm for generating an acceptable sequence of models was given. This algorithm

appears to have limitations in its application and is discussed in more detail later in this Chapter.

A short discussion of assembly line simulation with stochastic variables is given by Arcus [1], and Moreno [16] discusses transient effects in production lines.

The work presented here is deterministic and has more in common with references [22] and [20] than with the stochastic treatment in references [1] and [16].

4.2 The problem

The circumstances in which the simulation of the assembly line is undertaken are now given.

An assembly-line balance will have been obtained by suitable application of a balance program (Chapter 3). The information from the balance that is transmitted to the simulation will be a statement of how much work is required in each station to complete each model. As an illustration let there be 4 stations and 3 models, with work requirements as shown in Table 1 below.

Table 1

Model number (m)	Station number (j)				Model work content
	1	2	3	4	
1	2.01	4.01	0	3.15	9.17
2	1.72	2.10	5.14	3.17	12.13
3	1.56	0	2.20	5.32	9.08

Table 1 shows that, for example, 5.14 man-minutes of work is required in station 3 for model 2. The work durations are taken as constant throughout the whole of the shift, and their sum for each model over every station is model work content. The station work requirement is obtained by first multiplying each entry in a station column by the number required of the relevant model. Then the sum of the products so obtained for each station is the station work requirement.

The number of stations present is always known, and the position and degree of completion of work of product units already on the line is known. In balancing the shift requirement, the stated requirement for the next shift is used. In practice, the days production will comprise the following:-

(i) completion ofthe assembly of units not completed in the preceding shift,

(ii) complete assembly of the major part of the shift requirement,

(iii) partial assembly of the remainder of the shift requirement.

Details of (i) above are given as initial values of the problem, but (ii) and (iii) cannot be separated unless the sequence of products is known for the next shift. As

explained below, this sequence is part of the problem that is to be solved by the simulation. In practice the work content of (i) and of (iii) above will not differ greatly from each other, and the results show that it is sufficiently accurate to allocate work on the basis of the requirement of the next shift alone. In the simulation it is important to take note of the models that are already part assembled at start of shift.

Station passage time, and upstream and downstream allowance times (see 4.1.1 for definitions) are known, and launch interval is obtainable as the quotient of shift duration divided by units required in the shift as determined by the balance program (see Chapter 3.8). The simulation is carried out in one of the following 3 alternative situations:-

(i) The sequence of product models for the shift is known.

(ii) The shift requirement is known and the simulator is required to generate the complete sequence of product models.

(iii) The shift requirement is known and a sequence is given. The problem is to make rearrangements within this sequence in conformity with certain constraints.

A solution may be required in circumstances both where concurrency of operations (Chapter 2.2) is permitted or where it is not.

4.3 The mathematical model

4.3.1 Notation

In a shift of duration T minutes n product units p_i, (i = 1,2,...,n) made up from q different models where model type is defined by m, (m = 1,2,...,q) are passed in turn through λ stations S_j, (j = 1,2,...,λ). The passage time of a product through station j is t_j, and $t(u)_j$ and $t(d)_j$ are the upstream and downstream allowance times of station j (see 4.1.1 for definitions).

Each product must enter and leave each station. We write a_{ij}, x_{ij} to denote the times at which entry and exit respectively take place for product i at station j. Also for each product i the work allocated to station j must be started and ended, and s_{ij} and e_{ij} denote the relevant start-work and end-work times respectively.

The duration of the work allocated to station j for model m is d_{mj}.

The times at which events take place are reckoned from the instant at which the assembly line starts up for the shift. This is time zero.

4.3.2 Entry and exit relations

Entry to station j+1 is assumed to occur simultaneously with exit from station j. Also exit from station j occurs at a time t_j after entry to that station. We have therefore

$$x_{ij} = a_{ij} + t_j, \text{ for all products i and all stations j} \quad \ldots (1)$$

and $$x_{ij} = a_{i,j+1} \quad \ldots (2)$$

Let the number of uncompleted products at the start of shift be r. The position on the line of each of these products is specified by giving the exit time (relative to start of shift) of each from the station in which it is located at start of shift. The last uncompleted product that entered the line will be located in station 1. Therefore x_{r1} is known. Then

$$a_{r1} = x_{r1} - t_1 \quad \ldots (3)$$

Now the products are launched at equal intervals c, and therefore

$$a_{i+1,j} = a_{ij} + c \quad \ldots (4)$$

and from eq. (3) and (4) the entry time to the line of unit (r+1), the first unit to enter the line during the shift is

$$a_{r+1,1} = a_{r1} + c = x_{r1} - t_1 + c \quad \ldots (5)$$

All the quantities on the right of eq. (5) are given in the initial conditions and $a_{r+1,1}$ is therefore known. But with this knowledge eq. (1) and (2) permit every exit and entry time of product unit (r+1) to be determined. Furthermore eq. (4) permits every entry time of product unit (i+1) to be determined from corresponding entry times of product unit i.

In short, eq. (1) - (5) permit the exit and entry time of every product unit to be determined. These times are in no way affected by what work is done.

4.3.3 Times for starting and ending work

4.3.3.1 Situations where concurrent work is permitted

Consider station j. If concurrent work is permitted the operator in station j will be independent of the work situation in station (j-1). The rules are as follows:-

(i) Operator j may not start work on unit i before he has completed unit (i-1) or before the time $(a_{ij} - t(u)_j)$, the time at which unit i crosses the upstream allowance limit of station j.

(ii) Operator j will cease work on unit i at the time which is the smaller of $(s_{ij} + d_{mj})$ and $x_{ij} + t(d)_j$. That is, he will be able to finish the tasks by working the amount d_{mj} appropriate

to the product model concerned, unless the product unit has crossed the downstream allowance limit at which point he must at once stop and transfer to the next unit.

(iii) If the conditions of (i) above are met, the operator is assumed to move from one unit to the next without delay.

Let D_{ij}, I_{ij}, C_{ij} and U_{ij} be respectively the deficiency work, operator idle time, congestion and utility work (see 4.1.1 for the meaning of these terms) for product unit i in station j. Rules (i), (ii) and (iii) above are now applied to obtain relationships for D_{ij}, I_{ij}, C_{ij} and U_{ij} and for the start-work time s_{ij} (see 4.3.1).

Consider station j. Work on unit (i-1) will have ceased at $e_{i-1,j}$. This time may occur before unit i has crossed the upstream allowance limit of station j, or while unit i is within the region of the upstream allowance of j, or after unit i has entered station j. The upstream equations differ in these three instances as follows:-

$$0 \leq t(u)_j \leq a_{ij} - e_{i-1,j}; \quad \begin{cases} I_{ij} = a_{ij} - e_{i-1,j} - t(u)_j \\ D_{ij} = t(u)_j * \\ s_{ij} = a_{ij} - t(u)_j \end{cases} \quad \ldots (6)$$

$$a_{ij} - e_{i-1,j} \leq t(u)_j; \quad \begin{cases} I_{ij} = 0 \\ D_{ij} = a_{ij} - e_{i-1,j} \\ s_{ij} = e_{i-1,j} \end{cases} \quad \ldots (7)$$

$$a_{ij} \leq e_{i-1,j}; \quad \begin{cases} I_{ij} = D_{ij} = 0 \\ s_{ij} = e_{i-1,j} \end{cases} \quad \ldots (8)$$

With the start-work time s_{ij} known we may now develop further equations for the downstream situation. Again, there are three possible situations, concerning in this instance end-work time. They are that work may be incomplete when the unit passes the downstream allowance limit of the station, or work may finish within the downstream allowance region, or within the station itself. The downstream

---

* To maintain proper dimensionality equations of this form should be multiplied by the factor (1 assembly operator). Provided that it is specified that there is one operator per station (as has been done here, 4.1.1 (vii)) it is considered both permissible and clearer to omit the factor given above.

equations corresponding to these situations are

$$s_{ij} + d_{mj} \geq x_{ij} + t(d)_j, \quad \begin{cases} U_{ij} = s_{ij} + d_{mj} - x_{ij} - t(d)_j \\ C_{ij} = t(d)_j \\ e_{ij} = x_{ij} + t(d)_j \end{cases} \quad \ldots (9)$$

$$0 \leq e_{ij} - x_{ij} \leq t(d)_j, \quad \begin{cases} U_{ij} = 0 \\ C_{ij} = e_{ij} - x_{ij} \\ e_{ij} = s_{ij} + d_{mj} \end{cases} \quad \ldots (10)$$

$$e_{ij} \leq x_{ij}, \quad \begin{cases} U_{ij} = C_{ij} = 0 \\ e_{ij} = s_{ij} + d_{mj} \end{cases} \quad \ldots (11)$$

4.3.3.2 Situations when concurrent work is not permissible

When it is not permissible for two operators to work on one product unit, the rules for conduct of work are the same as those given in 4.3.3.1, (i), (ii) and (iii), but in addition a fourth rule is added thus:-

(iv) An operator in station j (j > 1) may not start work in unit i until the operator in station (j-1) has completed his task on this same unit i.

There are two relevant situations with respect to the operator in station (j-1). First, he is still working on

unit i when it enters station j. Second, he completes work on i in the upstream allowance region of station j. The upstream equations for station j, where j > 1, that correspond to these situations are:-

$e_{i,j-1} \geq a_{ij}$, where $e_{i,j-1} > e_{i-1,j}$

we have,
$$\begin{cases} I_{ij} = e_{i,j-1} - e_{i-1,j} \\ D_{ij} = 0 \\ s_{ij} = e_{i,j-1} \end{cases} \quad \ldots (12)$$

$a_{ij} \geq e_{i,j-1} \geq a_{ij} - t(u)_j$, where $e_{i,j-1} > e_{i-1,j}$

we have,
$$\begin{cases} I_{ij} = e_{i,j-1} - e_{i-1,j} \\ D_{ij} = a_{ij} - e_{i,j-1} \\ s_{ij} = e_{i,j-1} \end{cases} \quad \ldots (13)$$

4.3.4 Method of computation

The sets of eq. (6) - (13) given above provide recurrence relations for start-work and end-work times in all the relevant situations of the simulation, and permit computation of the station penalty work parameters D_{ij}, I_{ij}, C_{ij} and U_{ij} for each product unit in every station in terms of start-work and end-work times, provided that suitable initial values are available to allow the recurrence relations to be initiated. These initial values

are given as the end-work times in the station in which each unit is, for all the units left on the line from the previous shift. Consider the uncompleted product furthest down the line, with end of work time $e_{1\lambda}$ known. Consider that the next product is to be found in station $(\lambda-1)$. Then end of work time $e_{2,\lambda-1}$ will be an initial condition. With $e_{1\lambda}$ and $e_{2,\lambda-1}$ known the start-work time of product 2 in station λ is known. In this way the line is started up. The quantities that must be known at station j to apply the recurrence relations to product i are:-

the end-work time of the preceding product (product i-1) in station j, $e_{i-1,j}$

the end-work time of the current product (product i) in the preceding station $e_{i,j-1}$

the station passage time t_j, the station entry and exit times a_{ij} and x_{ij}, the task duration d_{mj}, and the station allowances $t(u)_j$ and $t(d)_j$.

Of these t_j, $t(u)_j$, $t(d)_j$, d_{mj} are known

a_{ij}, x_{ij} are obtainable without difficulty from eq. (1) - (5) and are model-sequence independent as already explained.

The only quantities that present difficulty are $e_{i-1,j}$ and $e_{i,j-1}$. However, let us compute the progress of product

(i-1) through all its stations and keep all the $e_{i-1,j}$. Then for product i, provided that we start at station 1 and move through the stations in their order in the assembly line, the $e_{i-1,j}$ will all be known. As the end-work times e_{ij} for product i are computed they are placed in the position held formerly by the $e_{i-1,j}$ and will then be available for later computations made to determine the $e_{i+1,j}$. The quantities $e_{i,j-1}$ required for computations for product i in station j are of course available from the preceding station, station (j-1). This method is suitably economical in use of computer core store.

Special computations are needed at the termination of the shift. In effecting these within the general framework of the method given above, several detailed problems required to be solved. Considerable complexity in the terminal logic was accepted in order that an exact overall check could be placed on the computation. This was done by computing total effort applied in two ways as follows.

total effort = (number of assembly stations) ×
(shift duration)

total effort = (running total of task durations) -
(running total of utility work) +
(running total of idle time)

If these two numbers are not equal there is an error in the computation, which can be identified. The check has been of great value in drawing attention to obscure peculiarities of the logic of the simulation.

The main output of the simulation comprises the aggregated idle time I, utility time U, work deficiency D and congestion C where

$$I = \sum_{i=1}^{n} \sum_{j=1}^{\lambda} I_{ij};$$

n is the number of product units handled in the shift

λ is the number of stations

and U, D and C are obtained from U_{ij}, C_{ij} and D_{ij} by similar computations. Station subtotals I_j, U_j, C_j, D_j are also recorded where

$I_j = \sum_{i=1}^{n} I_{ij}$, and U_j, C_j, D_j are given by similar relations.

Various other detailed computations are made that have special uses in the conduct of the simulator runs but are not of general interest.

The model discussed above is capable of carrying out a complete computation for a whole shift provided that the sequence is known and is not required to be altered.

If alterations to the sequence are required special algorithms that are included in the program must be used. These algorithms are included in the discussion of the simulation program which is now presented.

4.4 The computer simulation program

The main features of the computer simulation program are shown in figure 12.

The parts of the program that are of particular interest have in the main been described in 4.3 above. Thus, the blocks marked A_1 and A_2 in figure 12 are the application of eq. (1) - (5) above, and blocks marked B_1 and B_2 are the application of the sets of eq. (6) - (13) above. Block C is a fairly extensive set of statements that contains the logic for terminating the computation at a specified or computed shift termination time.

The other parts of the program that are of special interest are the sequencing algorithms. Initially, the usefulness of simple algorithms based on averages was assessed. However, these were found to be wholly unsatisfactory in instances where unevenness in allocation of tasks to stations was present. The final algorithms chosen use the same basic principle as the single algorithm used by Thomopoulos [19, 20], but differ significantly from this

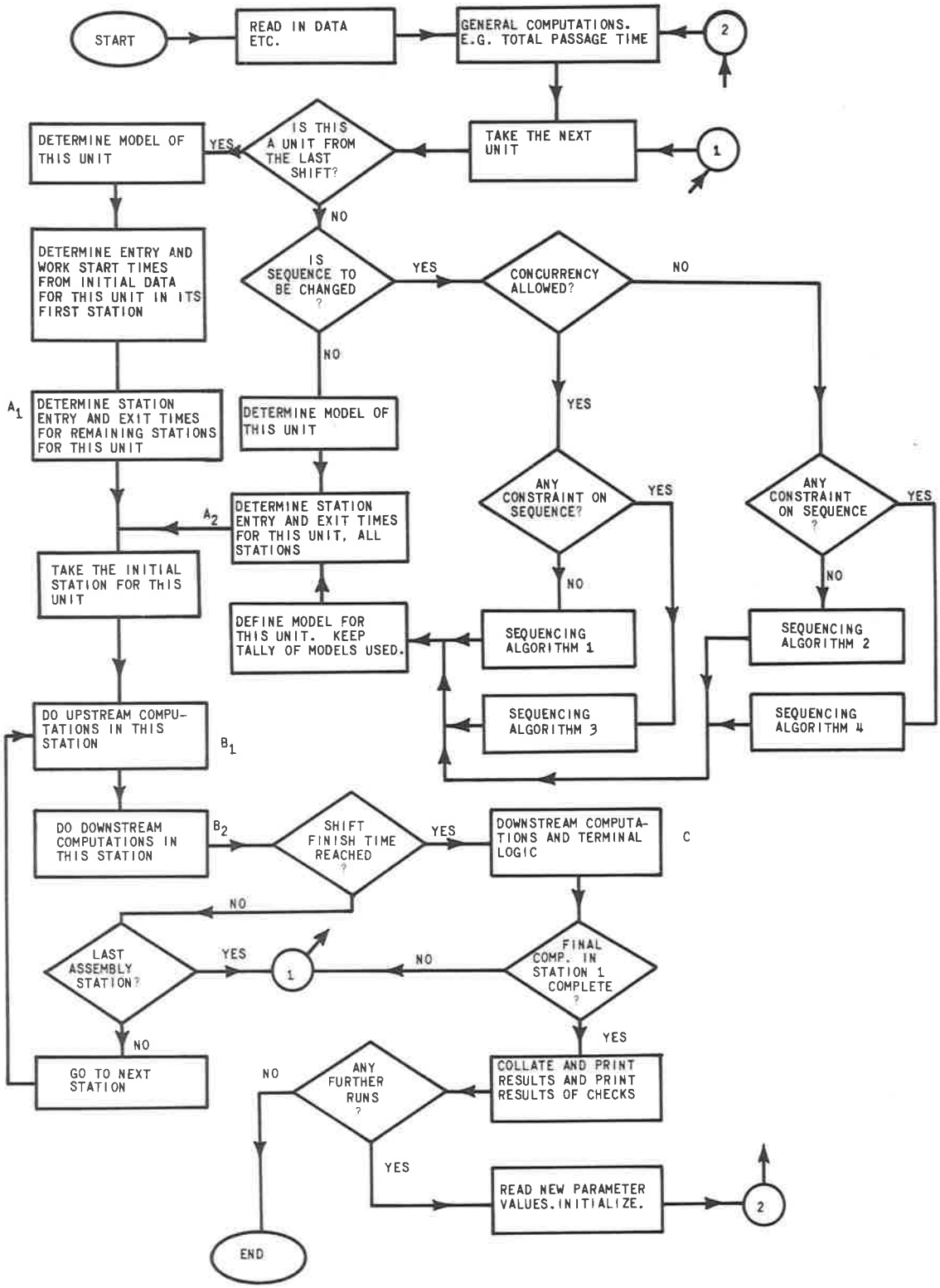


FIGURE 12 : GENERAL FLOW DIAGRAM OF COMPUTER SIMULATION PROGRAM FOR ASSEMBLY LINE

algorithm in important respects.

Thomopoulos' algorithm determines for each remaining model in every station the parameters work deficiency, idle time, congestion and utility work. Then, a penalty cost related to the type of work in each station is associated with each of these parameters, and a total penalty cost is worked out for each model. Finally the model with lowest penalty cost is selected as the next in the sequence. Because the penalty costs are related to actual costs, the parameters are not formulated in a way that facilitates arbitrary parameter changes designed to improve the selections made. Further, the procedure of always selecting the model with the lowest penalty cost appears to leave a residue of models with high penalty costs, that become increasingly difficult to locate in the sequence as the shift termination approaches.

Four algorithms are included in the simulation program described here to meet the situations of constrained or unconstrained sequencing in conditions where concurrence of work is permitted and is not permitted. The algorithms are described in turn below.

4.4.1 Sequencing algorithm 1

Sequencing algorithm 1(see figure 12) applies when

concurrent working is permitted and when there are no constraints on the positioning of models in the sequence. The algorithm (in common with the other three sequencing algorithms) starts with knowledge of the end-work time in every station for the preceding unit, and with knowledge of station entry and exit times of the current product, together with station data and model task durations. The objective of the algorithm is to select the model for the next product to be assembled. The distribution of models in the production requirement is given. (The order of products already on the line at the start of shift will not, of course, be changed.) The algorithm operates as follows:-

(1) Measures are taken to avoid use of models in excess of the requirement.

(2) U_{mj} and C_{mj} signify the utility work and congestion for the current unit in station j if this current unit were model m. When concurrent operations are permitted, the idle time and work deficiency associated with the current unit are brought about by the end-work time of the preceding unit and station entry and upstream allowance times of the current station. The model we choose now has no effect on the idle time associated with the current unit, but, by the argument just given,

does affect idle time and work deficiency for the next unit. If we regard I_{mj} and D_{mj} as being associated with product i, then I'_{mj} and D'_{mj} are associated with product (i+1), and we use I'_{mj} and D'_{mj} as parameters for the selection of the model of the ith product unit in the sequence.

The I'_{mj}, D'_{mj}, C_{mj}, U_{mj} are computed for each station then summed, so that an aggregate figure is obtained for each model, e.g.

$$I'_m = \sum_{j=1}^{\lambda} I'_{mj},$$

and U_m, C_m, D_m are determined in a corresponding way. Next, a criterion L_m is developed for each model where

$$L_m = aI'_m + bD'_m + cC_m + dU_m \qquad \ldots (14)$$

The parameters (a, b, c, d) may be varied arbitrarily. Typical values are (5, 2, 1, 20). The degree to which each of I'_m, D'_m, C_m, U_m influence L_m is dependent on these values.

(3) Two procedures (i), (ii) are used as follows.

(i) Set a switch that chooses procedure (ii) for the next product unit. Of these models that remain and for which $U_m = 0$ select the one

that has the largest work content. If no remaining model has $U_m = 0$, select that which has the lowest work content. Continue with the simulation for the chosen unit.

(ii) Set a switch that chooses procedure (i) above for the next product unit. Choose the model for which L_m is smallest.

The algorithm just described has been developed by trial and error. The philosophy is to obtain a sequence that is consistent with the values chosen for the parameters a, b, c, d. For example with the typical values (5, 1, 2, 20) great emphasis is placed on avoiding utility work. Idle time is held to be objectionable but much more acceptable than utility work. Congestion and work deficiency are mildly objected to. Although congestion is, here, preferred to work deficiency, it is given a larger value in order to bias the assembly work away from the region of utility work. This philosophy is arbitrarily chosen. In situations where it is unsuitable it can be remedied by changing the values of a, b, c, d as required. (Thomopoulos [20] uses the costs of the various types of penalty work for a, b, c and d, as discussed in 4.4 above.)

The alternate choice of a task of high work content

followed by one with low value of L_m has the recommendation that it works. (Utility work is kept down by the condition $U_m = 0$.) Algorithms were tried which selected the model that at each selection minimised L_m. This had the disadvantage that it tended to keep the low and high work content models until last. This frequently resulted in a disastrously bad latter end of the sequence.

4.4.2 Sequencing algorithm 2

Sequencing algorithm 2 (see figure 12) applies when concurrent working is <u>not</u> permitted and when there are no constraints on the placing of models in the sequence. The algorithm differs from algorithm 1 specifically as follows.

(i) A different significance is given to idle time in L_m. For a situation of no concurrency of work, the most serious cause of idle time for product i in station j is failure by the operator in station (j-1) to finish his work on product i in time to avoid causing the operator in station j to wait for him. We therefore write

$$L_m = aI''_m + bD''_m + cC_m + dU_m$$

and compute I''_m and D''_m by means of the standard upstream equations for the current product (and

not for the next product as in algorithm 1). It might be more logical to take account of I'_m and D'_m as defined for algorithm 1 as well. However the value of these compared with I''_m is, in general, small and they have been neglected.

(ii) In the procedure that corresponds to procedure 3(ii) of algorithm (i), an additional condition of selection is applied. This is that the value of I''_m must be less than some arbitrarily chosen value. An example of a typical value is 1.30 man minutes of idle time.

4.4.3 Sequencing algorithm 3

Sequencing algorithm 3 applies when concurrent work is permitted but there are constraints on the sequencing as follows. Assume that the assembly line being sequenced is a subsidiary line operating in parallel with a main assembly line to which it must supply assembled units for incorporation in the product of the main assembly line. Launch interval on both lines is the same. There is latitude in the system, however, because the subsidiary line is shorter than the main line, in such a way that if product i is completed now, (r-1) other products may be completed before it is necessary to load i into the product of the main line (e.g. car front seats are assembled on

the subsidiary line and car bodies on the main line). r is called the stack size. Also, the sequence of models in the main line is known only as far back as $(v+\lambda+r)$ stations from the point of loading in the main line. λ is the number of stations in the subsidiary line: v is called the look ahead. As soon as a launch interval is complete a unit must be loaded onto the subsidiary line. Thus, for example, irrespective of stack size, if look ahead is 1, there will be no choice in what can be loaded because only one product is known (random selections are not permitted). Equally, irrespective of size of look ahead, if stack size is 1 there will be no opportunity to complete any other product before the product being loaded onto the subsidiary l: must be loaded into the main line product. The objective of the sequencing algorithm 3 is to set up a suitable sequence from an existing sequence in the conditions given above. The existing sequence represents the main line sequence and the alterations represent changes from this sequence which will both be permissible and will facilitate assembly on the subsidiary line. The procedure is as follows:-

(1) A sequence of products for the shift is given as part of the initial values of the problem. From this sequence a list of length v called the look ahead list

is set up and each member is given a credit equal to r. Any member of the list v may be chosen to be next onto the line, but when this is done the credit of every member of v that initially preceded it must be reduced by 1. Before any member of the list may enter the assembly line the credit of the first member of the list must be checked. If it is 1, then that member must be next on line. If credit is greater than 1 the selection procedure detailed below is used.

(2) L_m is computed by the same procedure as in sequencing algorithm 1.

(3) Two procedures (i) and (ii) are used as follows

(i) A switch is set in such a way that procedure (ii) below is used for the next unit. Then, an example of each model present among members of the look ahead list is considered and the first example of the model for which L_m is smallest is selected.

(ii) A switch is set in such a way that procedure (i) above is used for the next unit. Next, if the look ahead list contains an example of the greatest-work-content model, and if $U_m = 0$, then the first example of that model is

selected. If the foregoing conditions cannot be met the first example of the model in the look ahead list that has the greatest work content of the models in the list is selected.

4.4.4 Sequencing algorithm 4

Sequencing algorithm 4 applies when concurrent work is not permitted and when there are constraints as in 4.4.3 above on the sequencing. The algorithm is the same as sequencing algorithm 3 except in relation to the following:-

(i) L_m is computed by the same procedure as in sequencing algorithm 2.

(ii) In the procedure that corresponds to procedure 3(ii) of algorithm 3, an additional condition of selection is applied. This is that the value of I''_m must be less than some arbitrarily chosen value, e.g. 1.30 man minutes of idle time

4.5 Simulation test runs

After the simulation program had been checked out by a series of short runs with hand checks some runs were carried out to assess the usefulness of the simulation and to investigate the performance of the sequencing algorithms. In order to keep the cost of the computer runs within bounds

without sacrificing all realism the running period was made equivalent to approximately $\frac{1}{3}$ ofan 8 hour shift, and a mix of 7 models requiring 9 assembly stations was used. Two sets of task allocations to stations by models were used and these are shown in Tables 1 and 2 of Appendix I. Both tables are artificially constructed. Table 1 is intended to represent the results of multiple individual balances that have been rather unsuccessful in terms of overall efficiency, e.g. model 1 has an overall efficiency of (2.05/2.25) = 0.91 and model 7 an overall efficiency of 0.94. (The allocations have been repeated in groups of three in this test in the hope that mechanisms of assembly performance might be more easily recognised.) Table 2 is intended to represent an aggregated task group balance of high efficiency, but in which allocations are particularly uneven e.g. model 1 station 3 has a zero requirement for assembly work.

For each input, station-passage time has been made to correspond to the largest entry in the relevant Table, namely 3.45 min. for Table 1 and 4.34 min. for Table 2. For Table 2 this gives an assembly-line length of 39.06 min. against a work content of 29.38 min. Results of simulator tests are now discussed.

4.5.1 Simulator test 1

Simulator test 1 was carried out with the main aim of investigating the performance of sequencing algorithms 1 and 2 in easy circumstances. Subsidiary aims were to investigate the effect of upstream and downstream allowances on performance, to compare the effect of including and excluding concurrent work and to examine the effect of changes in parameters b and c in L_m. The values of work deficiency, idle time, utility work and congestion given in the Tables of Appendix I are the aggregated values summed for every product unit over every station (see 4.3.4 above).

The main results shown by the runs are that

(i) Both sequencing algorithms operated effectively (see Table 3, Appendix I). Utility work was held to a trivial level, and total operator idle time ranged between about 6 man minutes and 33 man minutes in a shift of some 1420 man minutes of effort. Two runs, 01 and 02 in Table 3, Appendix I, have been done with the hand-prepared sequence shown in Table 4, Appendix I, Sequence 1. This sequence has been arranged so that the models are evenly distributed. This type of arrangement has been found to be

associated with good assembly performance. Run 7 Table 3 Appendix I is comparable with run 01 and shows much improved idle time results. Run 13 Table 3, Appendix I is comparable with run 02 and performance is similar but slightly worse.

(ii) As expected, provided that the sequence of units is adequate, the shift efficiency for the mixed assembly is much greater than that of the best of the individual balances of Table 1 Appendix I e.g. run 1 Table 3 has an efficiency of (1423.08 - 5.54)/(1423.08) which is 0.996 approximately - an extremely high efficiency against 0.94 in Table 1.

(iii) Operator idle time is insensitive to changes in upstream and downstream allowance. Work deficiency (as would be expected) decreases sharply with decrease in upstream allowance.

(iv) Small alterations in the L_m parameters b and c had little effect. Compare runs 1 to 6 and 7 to 12 in Table 3, Appendix I.

(v) Removal of the facility for concurrent work by operators had a most significant effect.

Compare for example in Table 3, Appendix I, the idle time of runs 13, 14, 15 (31.76, 31.36, 31.16 man minutes) with the idle time of runs 7, 8, 9 (7.03, 7.03, 7.13 man minutes). These runs differ only in that work concurrency is excluded for runs 13, 14, 15 and permitted in runs 7, 8, 9.

(vi) Work congestion increases significantly with increase in downstream allowance (as would be expected) and also increases significantly with decrease in upstream allowance. This latter mechanism is also to be expected. If, for example, three consecutive tasks in station j result in x units of work deficiency on the first and y units of congestion on the third, then if the upstream allowance time is reduced by 0.5 minutes, the congestion will be increased by 0.5 man minutes provided that there is sufficient downstream allowance time to permit the increase (if downstream allowance time is insufficient, utility work will appear instead).

(vii) Sequence 2, Table 4, Appendix I, is an example of a sequence generated by algorithm 2 in the fairly difficult circumstances of run 18. The

initial alternation of high and low work content models is apparent in the selection pattern. The sequence is satisfactory.

4.5.2 Simulator test 2

Simulator test 2 was carried out with the same general intention as for simulator test 1 described above, and the results were substantially similar. Note however that

(i) The uneven allocations of test input 2 (Table 2, Appendix I) presented a more difficult problem for the sequencing algorithms to handle than did test input 1. In particular, the idle time for the runs in Table 5 AppendixI in which concurrent work was permitted (runs 1 to 6) had values within the range 25 man minutes to 42 man minutes. In the comparable runs in simulator test 1 (runs 7 to 12 Table 3) the maximum idle time recorded was 8.53 man minutes.

(ii) A tendency is shown for idle time to increase as downstream allowance is reduced. Because the sequencing algorithms avoid the incidence of utility work it follows that if downstream allowance is reduced congestion will be reduced also as is very noticeably the case in runs

1, 2, 3 Table 5, Appendix I for which the increase in idle time just noted is greatest. As a result the operations generally will tend to take place more towards the upstream region of the stations, and in certain stations there will be a vulnerability to idle time.

(iii) Runs 01 and 02 of Table 5, Appendix I are obtained from the evenly distributed hand-prepared sequence 1, Table 6, Appendix I. The comparable sequenced runs are runs 1 and 7 Table 5, Appendix I. Performance is similar with the advantage sometime with the hand-prepared sequence except in utility work where the automated results are significantly better. Sequence 2 Table 6, Appendix I shows the sequence generated for run 10 Table 5, Appendix I. The sequence shows some bunching of residual high work content models (units 73 to 77).

The performance of the algorithms although less good than in the comparable instance in test 1 is nevertheless more than adequate. The greatest idle time is less than 4% of the effort supplied, and utility work at about 5 man minutes maximum is negligible.

The result is most encouraging as it suggests that the very uneven task allocations typical of aggregated task group balances will not pose problems that are too difficult for an automatic sequencing algorithm to solve.

4.5.3 Simulator test 3

The aim of simulator test 3 was to consider the performance of sequencing algorithms 3 and 4. The method was as follows. The model-mix requirement was used as the basis of a hand produced sequence of products intended to give poor results. This was done by bunching products of high work content and of low work content in groups. The sequence used with test input 1 is shown in sequence 1, Table 8, Appendix I, in which it should be noted that numbers have been allocated to models in ascending order of model work-content.

Simulator runs were carried out for stack sizes with values 4, 6 and 8 and with look ahead lying between 5 and 16. (Stack size and look ahead are defined and discussed in 4.4.3 above.) Upstream and downstream allowance times were maintained at 0.50 min. and 1.25 min. throughout the runs of the experiment. The criterion for idle time described in section 4.4.4 (ii) above was set at 1.30 man minutes throughout.

A typical set of results is shown in figure 13 for test input 1 (Table 1, Appendix I) with concurrent working permitted. Table 7 shows the most significant results for input 1 with concurrent work excluded and for input 2 (Table 2, Appendix I) in conditions of inclusion and exclusion of concurrent work. The performance in runs 1 to 9 in Table 7 are the results of the operation of sequencing algorithm 3 on this sequence. Run 02 is the result when sequence 1 is applied to the assembly line and is not modified. The results in run 9 (and in the other comparable instances) are a significant improvement on the corresponding unsequenced run, run 02.

(i) In all instances investigated algorithms 3 and 4 kept the idle time and utility work within bounds. The greatest value of idle time was 49.31 man minutes and of utility time was 25.00 man minutes.

(ii) Figure 13 shows clearly that idle time tends to decrease with increase in stack size, and this tendency is also evident in Table 7 for look ahead values of 8 and 16. When look ahead is less than stack size the advantage of increased stack size tends to be lost (as would be expected).

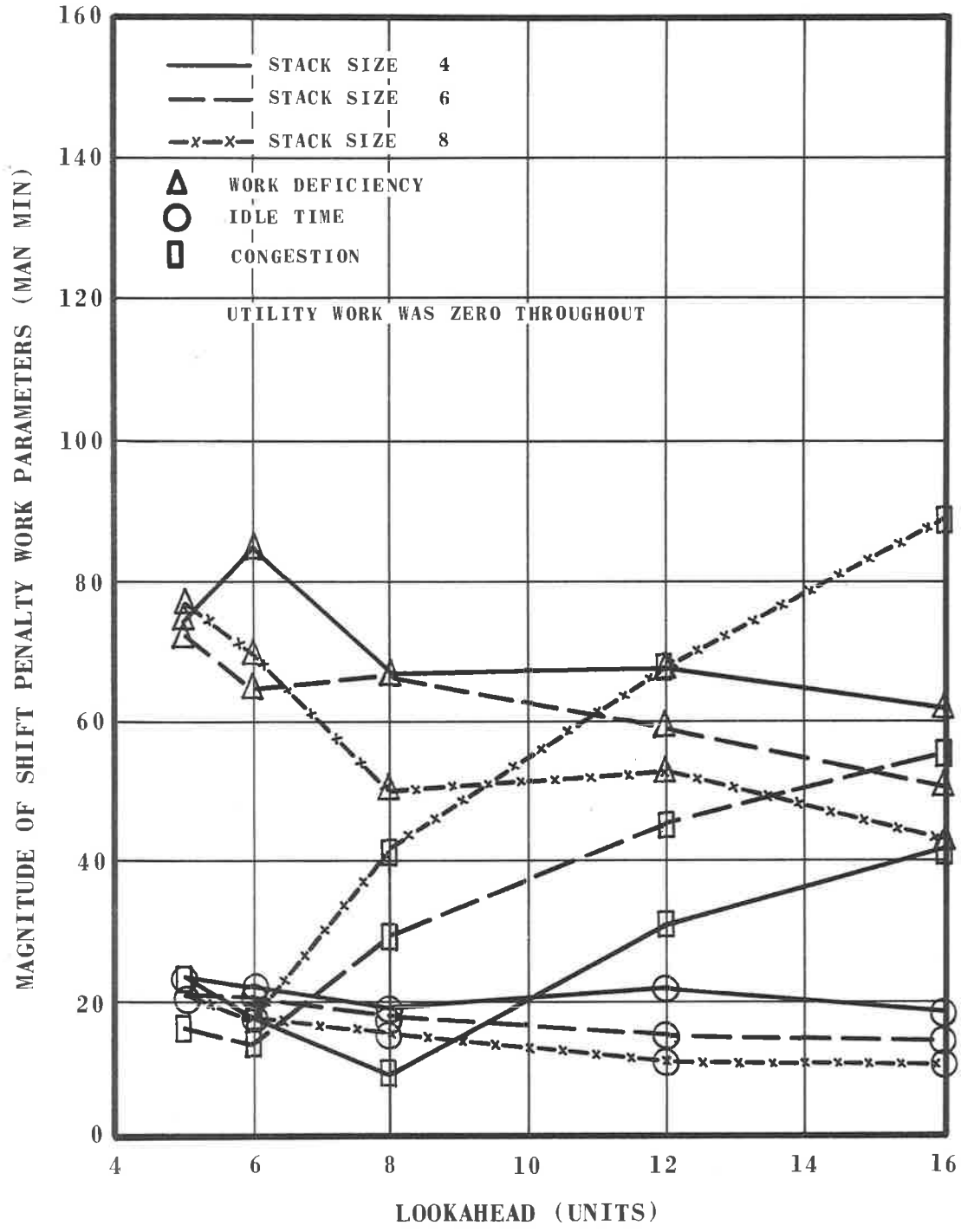


FIGURE 13 : PERFORMANCE OF SEQUENCING ALGORITHM 3 PLOTTED AGAINST LOOKAHEAD FOR DIFFERENT VALUES OF STACK SIZE

(iii) Sequence 2, Table 8, Appendix I shows the sequence achieved by sequencing algorithm 4 in run 9 of Table 7. The absence of bunching is associated with the fact the the "look ahead list" is maintained at a constant value throughout (i.e. it was not tapered off to zero members in the concluding stages of the simulation).

(iv) For test input 1 there was no requirement for utility work. For test input 2 there was some tendency for utility work to decrease with increase in stack size. The trend was not a strong one nor present everywhere.

(v) For a given input with a given state of acceptance or prohibition of concurrent work the results were comparatively slow changing and appeared predictable. This suggests that in a given situation the performance of a sequencing algorithm could be significantly improved by detailed analysis of the effect of parameter changes on assembly-line performance.

Certain more general conclusions concerning the use of the simulator program are given in Chapter 6. These more

general matters are related to results of the practical application presented in Chapter 5 as well as to the simulator test results. It is therefore more appropriate to discuss them in Chapter 6 than here.

CHAPTER 5. PRACTICAL APPLICATION

5.1 General

In this Chapter the balance and simulator programs are applied to a practical assembly-line problem. The aim of this practical application is to prove the method in the environment of a real assembly line. In addition, however, the opportunity has been taken to investigate performance in a variety of realistic conditions and to reach conclusions, whenever possible, that have a general application.

5.2 The problem

The problem studied was the assembly of motor vehicle front seats on a small self-contained assembly line. Some 15 different models were involved in the assembly operation and included bench seats and bucket seats in various quantities suitable for private cars, pickup trucks and panel vans. The assembly was divided into two main parts. First, the spring structure of the seat required to be assembled, and was followed by the assembly of the padding and seat covers. In this latter part of the work, the elemental task durations tended to be quite large fractions of the launch interval of the product units.

The seats made on the assembly line required to be

fitted into car bodies that were being assembled concurrently on another (main) assembly line. This situation was the basis of the development of sequencing algorithms 3 and 4 discussed in Chapter 4. All the measurements and task durations used in this Chapter have been taken from the real situation. It has, however, for technical reasons not been possible to validate the simulation by direct comparison with the performance of the assembly line. Nevertheless, the main aim of the research is the investigation of a general method and therefore it is considered that the omission of validation in a specific situation can be accepted without significant harm to the research. This matter is discussed further in Chapter 6.

The work content of the different models used in the simulation are shown in Table 3, Appendix II. It will be noted that the first four models in Table 3 have the same work content. The reason for this is that certain work that is different in each of these four models is done by specialist workers not employed as general assembly operators. Despite the fact that these four models are, for the purposes of the present assembly operation, the same, they have been treated as separate models in order to provide a more stringent exercise.

The balance method used was the aggregated task-group technique (Chapter 3.3.1) and it was arbitrarily decided to balance to 10 stations. Balances were made with two different model mixes that were reasonably typical of a day's requirement on the assembly line. The relevant results obtained from these balances were then applied to the computer simulation and the performance of the shift was investigated with various parameter values. Finally a third balance was made in which methods of allocation of two-man tasks were investigated. The simulation program does not contain logic to cater for two-man stations, and therefore no simulator runs were made in this condition. Some of the implications of inclusion of two-man stations are discussed.

5.3 Application with first model-mix

5.3.1 Combined precedence graph

The first requirement was the preparation of a combined precedence graph (Chapter 2.3.2). This was done by hand and entailed the production of individual precedence graphs for each model. In combining these into one graph to suit the balance program the following conditions needed to be met:-

(i) Nodes required to be numbered so that they formed a complete sequence of the natural numbers

and so that no node had a number greater than any of its successors.

(ii) If the difference between the number of a node and of one of its immediate predecessors was greater than 30, a dummy node needed to be inserted.

(iii) Care was needed to ensure that elemental tasks common to more than one model were included only once.

(iv) Every precedence link present in the original precedence graphs had to be considered for inclusion in the combined graph.

It was found convenient to write a small edit program to check items (i) to (iii) above. The final combined graph contained 191 nodes including dummies and tasks set to zero duration. Part of this graph including the separator set (Chapter 3.14) and certain areas of special interest is shown in figure 14. Preparation of the graph in the form of a geometrical diagram (as opposed, for example, to sets of numbers) was a definite help in the formulation. Table 1, Appendix II shows all the elemental task durations associated with the combined precedence graphs. (The complete list of tasks has been included in order to give an idea of the

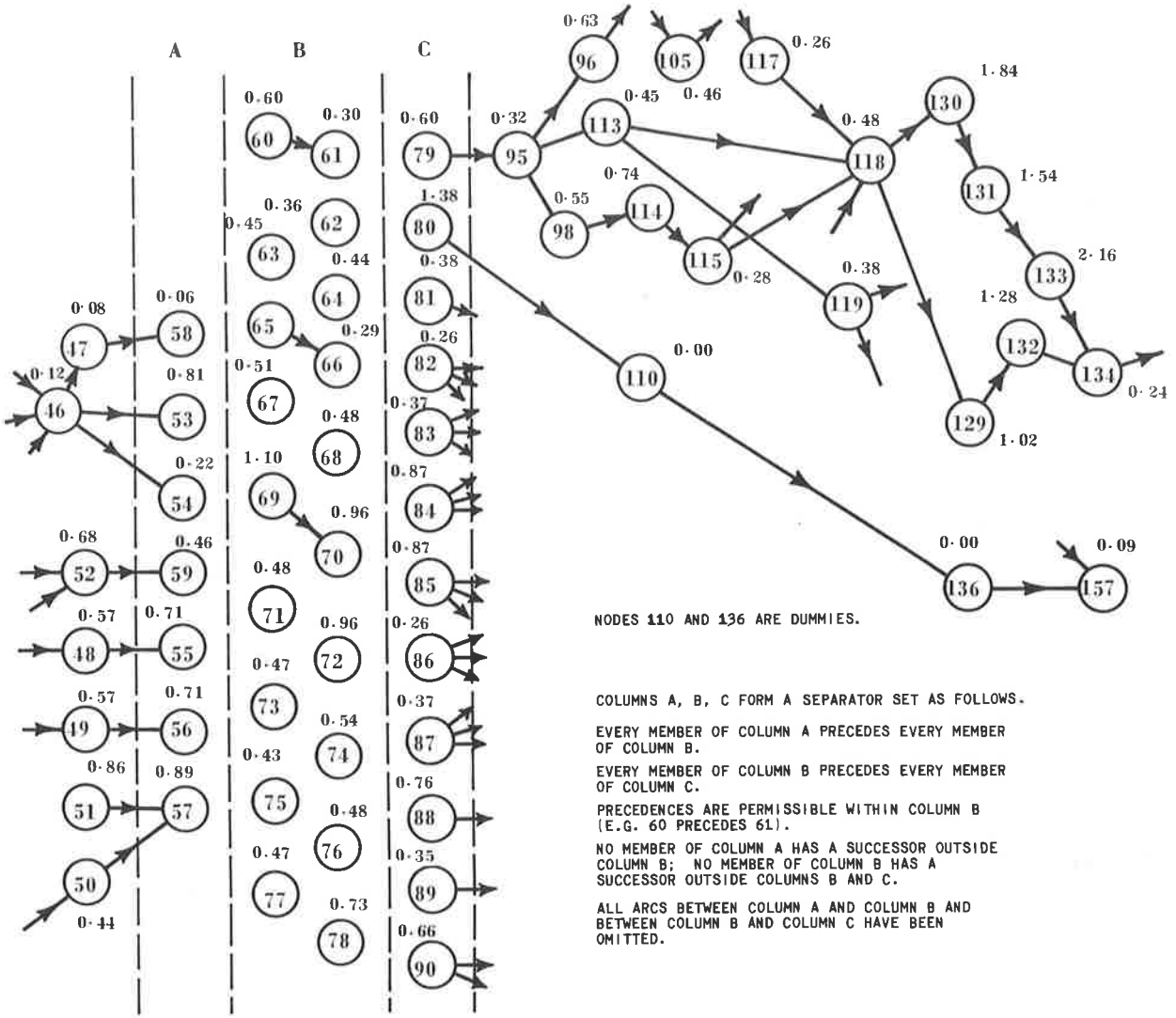


FIGURE 14 : PORTION OF COMBINED PRECEDENCE GRAPH

magnitude and distribution of the task durations.) Table 2, Appendix II illustrates some typical model identification vectors.

5.3.2 Computer inputs

The inputs to the balance program comprised the following:-

(i) Each node and its immediate successor nodes. These were read in one by one and the successors were at once written into the appropriate bit location of a word allocated to the parent node.

(ii) Task element duration associated with each node.

(iii) Production requirement by models.

(iv) Shift duration and number of stations.

(v) Model identification vectors.

5.3.3 The balance

The method of operation ofthe aggregated task group balance program has been discussed in detail in Chapter 3. Table 3, Appendix II shows the initial model mix. The initial work content of this mix was 4613.10 man minutes, and the effort available was 4500.00 man minutes (450 minute shift × 10 operators). The program reduced the

requirement by successively subtracting one unit of each model starting with that with the smallest work content. A balance was finally achieved after six units had been removed and the work content of the requirement was 4471.84 man minutes. The final model mix is also shown in Table 3 of Appendix II. The final work content added to the work content of the units removed should equal the initial work content. We have

$$4471.84 + 22.51 + 22.51 + 22.51 + 22.51 + 25.15 + 26.07 = 4613.10$$

and this is the initial work content as given above.

In achieving the balance all 191 nodes were handled by the program irrespective of whether they had a duration greater than zero or not. The method of splitting task groups when required as described in Chapter 3.3.1 was incorporated in the program, and experience with it is discussed below. The balance achieved is shown in Table 4, Appendix II, by station totals. The overall balance efficiency is very high at 0.994. This was achieved mainly on account of the large number of alternative tasks available owing to the nature of the precedence constraints, and only to a small extent because of task group splitting noted above. In fact only one split was present, between

stations 2 and 3. Relevant parts of the balance to illustrate this are shown in Table 5, Appendix II. The particular form of the splitting algorithm used in this version of the balance program operated as follows.

Consider the situation in station 2 as shown in Table 5, Appendix II. After allocation of task group 16 the algorithm reaches the end of the active list without being able to make a further allocation. Accordingly it takes the task group of minimum positional weight (i.e. the last group to be tested for allocation), which in this instance is task group 12. The adjusted value of task group magnitude is 6533 man minutes. (This value is obtained from the values computed after adjustment of model requirement before each balance iteration.) Station residue is (450.00 - 445.01) = 4.99 man minutes. The algorithm examines models in increasing order of work content. The first model is model 1 and there are 3 units of this model. Task element duration is 0.47 man minutes, and therefore the magnitude of effort required for these 3 units is 1.41 man minutes. 3 units of model 2 add 1.41 man minutes and 1 unit of model 3 adds 0.47 man minutes giving 3.29 man minutes all told. The 5 units of model 4 have a magnitude greater than (4.99 - 3.27) = 1.72 man minutes and therefore cannot be allocated to statio 2. After allocating 3.29 man minutes to station 2, the

program allocates the residue of task group 12 to station 3 and then proceeds with allocation to station 3 in the ordinary way.

The separator set, figure 14, operated effectively and allocated tasks 60, 65, 61, 66, 67, 64, 69, 72, 70, 63, 78 to the last 11 places in station 4 and tasks 62, 74, 76, 73, 77, 75, 71, 68 to the first eight places of station 5, thus separating columns A and C figure 14.

Table 6, Appendix II shows the allocations of work to stations by model as computed by the balance program, together with the station passage times allotted. The distribution of allocations of work is so uneven that it would be impractical to make all station passage times equal, as was done in the simulation tests. The assembly line passage time is 53.00 minutes, and the greatest model work content is 43.93 man minutes, giving a ratio 53.00/43.93 or 1.21. This ratio could have been reduced, but some latitude was considered desirable for the first application.

5.3.4 Simulator runs

Simulator runs similar to those made in the tests reported in Chapter 4, but less extensive, were then made, using the allocations of work shown in Table 6, Appendix II. A hand distributed sequence (intended to be reasonably

evenly distributed) was used for unsequenced runs and for sequencing under constraint. This sequence is sequence 1 of Table 7, Appendix II. The results of the simulator runs are given in Tables 7, 8 and 9 of Appendix II. Table 7 gives results of sequence generation, Table 8 deals with assembly performance in terms of work deficiency, idle time, utility work and congestion, and Table 9 shows the distribution of idle time and utility work for different runs over the three stations that showed on average, the highest values of these two quantities. In these Tables, runs 1 to 12 relate to unconstrained sequencing, with and without concurrent work; runs 13 to 18 relate to constrained sequencing, again with and without concurrent work. Run 19 is a simulation done with the sequence of models shown in Sequence 1, Table 7, Appendix II.

Comments on the results are as follows:-

(i) The sequencing algorithms have achieved stable results in all runs. Comparison of run 1 with run 19 shows that performance with the algorithm-generated sequence is superior to that with the hand-prepared sequence in regard to every parameter recorded in Table 8, Appendix II. The advantage is significant in regard to idle time but particularly significant in regard

to utility work where the comparison is 3.80 man minutes to 39.63 man minutes.

(ii) When utility work is present assessment of efficiency presents difficulty. If, as a guide, utility work is taken as equivalent to idle time for purposes of computing efficiency, then the efficiency of run 1 is about 0.985 and of run 11, the worst run, is about 0.925.

(iii) A marked advantage is obtained by permitting work to be done concurrently as shown by the results both for unconstrained sequencing (runs 1 to 12, Table 8, Appendix II) and for constrained sequencing (runs 13 to 18, Table 8, Appendix II).

(iv) The results with unconstrained sequencing show a considerable sensitivity to changes in upstream and downstream allowance times. The general trend is for performance to deteriorate as the sum of upstream and downstream allowance time is reduced. Run 12 Table 8 however, reverses this trend quite noticeably.

(v) The results with sequencing under constraint (runs 13 to 18, Table 8, Appendix II) show

insensitivity to changes in look ahead magnitude.

(vi) Sequence 2, Table 7, Appendix II is the automatically generated sequence associated with run 1, the run that gave the best performance in Table 8. The sequence seems good except for the bunching of 4 units of model 9 in the last four places in the sequence. The conditions that bring this bunching about are artificial, for the following reason. The sequencing algorithm has, at the start, a list of 151 units (169 required - 18 already on the line) to choose from. For the 151st selection this list has been reduced to a length of 1 unit. In practice it would probably be acceptable to permit a few of the models required in the next shift to be drawn on to provide good terminal conditions in the current shift. The sequencing under constraint has been arranged in this way (i.e. the look ahead list is kept full of units throughout the sequencing) and as a result better distribution is obtained at the end of the sequences than with the unconstrained sequencing. Sequence 5 of Table 7

Appendix II is an example of final models in a constrained sequence, while sequences 3 and 4 of the same Table are two of the worst examples of bunching for unconstrained sequences. If some choice can be given to the sequencing algorithms in unconstrained sequencing when the last few models are being selected, assembly efficiencies would be improved. The efficiency of the next shift would not be significantly reduced by this procedure.

(vii) Table 9, Appendix II shows the distribution of idle time and utility work over the three stations that, on average, had the highest values of these two quantities. The results are noteworthy, for they show clearly that the pattern of distribution of idle time and utility work is related to the sequencing method used. Thus the distributions in runs 1 to 6 done with sequencing algorithm 1 resemble each other, but show a sharp difference from those in runs 7 to 12 done with sequencing algorithm 2. The distributions in runs 13 to 15

done with sequencing algorithm 3 resemble closely the distributions of runs 16 to 18 done with the sequencing algorithm 4 but differ considerably from distributions in runs 1 to 6 and 6 to 12. Finally, the distribution associated with run 19, the hand-prepared sequence, is different again.

The importance of these results is that they show that, in one instance at least, the incidence of penalty work cannot be gauged by a study of the allocations of work to stations by models (e.g. Table 6, Appendix II). Knowledge of which stations will bear the brunt of the idle time and utility work will be of considerable value in the planning of assembly operations.

5.4 Application with second model-mix

5.4.1 The balance

The task durations, combined precedence graph and model identification vectors used were the same as for the first model-mix, except where specific statements are made to the contrary. The initial and final model-mixes are shown in Table 10, Appendix II and the balance achieved is shown in Table 11, Appendix II.

One of the aims in this balance application was to investigate the possibility of smoothing the allocation of work to stations by manipulation of the precedence graph through changes in the positional weights. An unsatisfactory feature of the first balance was the allocation of 9.16 man minutes of work to station 6 for model 15 (Table 6, Appendix II). Analysis showed that the cause of this was the allocation to station 6 of task groups associated with the elemental tasks 130, 131 and 133 (see figure 14). In order to investigate methods of modifying this situation it was assumed that it would be physically feasible to rearrange the method of manufacture so that task 133 became a task with no successors (the engineering feasibility of doing this was not taken into account). This entailed the removal of the arc (133, 134) in figure 14. Next, the positional weights of nodes 130, 131 and 133 were arbitrarily changed with the aim of locating them in station 9. In the balance, the desired relocation was achieved, and the allocation of work by models to stations is shown in Table 12 which also shows station passage times allotted. The allocation of work is noticeably more even than with the first model-mix. Also, the latitude (c.f. Table 6, Appendix II) in station passage times has been, in effect, removed. The total assembly passage time was 50.50 and the ratio of total passage time to greatest work content

was 1.14. The balance achieved is shown in Table 11. The efficiency was 0.989, an excellent result, which was achieved without any task-group splitting.

5.4.2 Simulator runs

Table 13, Appendix II shows the results of 17 simulator runs with the second model-mix. The shift duration was 30 minutes greater than with the first model-mix. The results obtained are similar. The best runs for model-mix 1 (e.g. run 1 Table 8) were noticeably better than the best runs for model-mix 2 (e.g. run 1 Table 13). However, for the worst runs, model-mix 2 gave the better results (c.f. run 10 Table 13 and run 9 Table 8). The distribution of idle time and utility work over stations was more even with model-mix 2 (as would have been expected because of the more even allocation of work to stations) and did not exhibit the clearly recognizable patterns noted with model-mix 1 (Table 9, Appendix II).

The sequences generated by the sequencing algorithms were in general similar to those for model-mix 1, but there was less tendency for bunching of model 9 at the end of the sequences.

The most interesting result of the runs is the very great general similarity of performance of the assembly

line with a significantly different model-mix.

5.5 Application with two-man tasks

Certain tasks in the combined precedence graph are tasks that require to be done by two men working together. These tasks have purposely been treated to date as is if they had been one-man tasks, because the inclusion of two-man tasks would have obscured certain of the main issues being investigated. However, now that the main material on assembly-line performance has been obtained it is of interest to consider some of the implications of the presence of two-man tasks in an assembly operation.

If there are several two-man tasks it will be desirable to collect them all into as few stations as possible, since stations where such tasks occur will need to be manned by two men. If concurrency of work is not permitted, then a scattered allocation of two-man tasks will be particularly harmful to efficiency. Single-man tasks allocated to a two-man station in such circumstances bring with them a content of idle time equal to their own work content, for the second operator in the station must stand idle until they are complete. However, in these conditions of prohibition of work concurrency, a balance can be computed in the usual way after minor program modifications. Further, if no facility exists in the program,

allowance may be made for the two-man tasks by simple hand adjustments.

If concurrent working is always permitted the need to ensure the concentration of two-man tasks into as few stations as possible is not so pressing. However, special arrangements must be made in the balance program to allow for the presence in each two-man station of two operators both of whom may do single-man tasks simultaneously.

If certain combinations of tasks can be done concurrently and certain others not, a considerable degree of added complication must be introduced to the program if it is to solve the balance problem in a general way.

Here, the assumption has been made that no concurrent work is permissible and the following elemental tasks are declared two-men tasks: 118, 129, 132, 134, 138 to 148, 154, 157 to 164. The task duration on the precedence diagram now relates to two men - e.g. the duration 0.48 required for task 118 now signifies that two men are required for 0.48 minutes each.

Study of the combined precedence graph suggested that proper manning could be achieved solely by changes in group-positional weights, and the changes shown in Table 14

were made. The relevant parts of the balance achieved are shown in Table 15. As shown, the tasks spread over four stations, but the portion of task group 118 that is in station 6 may be moved to station 7 without infringing precedences. The content of station 7 would then be too great, but task group 120 may be pushed down into station 8 with impunity, and the allocation of all the two-man tasks to three stations is achieved.

Without inclusion of the changes in positional weights noted above the two-man tasks were distributed over 4 stations in a way that made hand adjustment impracticable. It is of interest to note that, for the circumstances of this example, if three double stations are sufficient, the total idleness increases from the ten single-station value of 50.50 man minutes to 449.82 man minutes. The overall balance efficiency drops from 0.989 to 0.928. If four double stations are necessary, the idleness becomes 929.82 man minutes and overall balance efficiency falls to 0.86.

CHAPTER 6. DISCUSSION

6.1 General

In the foregoing Chapters a method has been developed for the planning and control of assembly operations in a mixed-model environment. The method has been to allocate work associated with different models of a product to a group of operators, and then against a given requirement of models to generate automatically sequences of product units suitable for application to the assembly line.

In allocating the work two main alternatives are offered. On the one hand, the work of each model can be evenly divided between stations. This approach has the serious disadvantage that the same task present in different models may be allocated to different stations, calling for a proliferation of skills among assembly operators. The other alternative depends on collecting together from each model every example of some given task and allocating it in one packet to a given operator. This has the immediate disadvantage that it increases the length of the assembly line and hence increases investment in inventory. It can also produce a grossly uneven allocation of the work content of a given model to the stations. It was suspected that in the circumstances of these uneven allocations it would be a matter of great difficulty to

provide a product sequence that would permit the assembly to be even marginally efficient.

However, despite uneven allocation, it has been shown that with constant task durations, effective sequences may be developed. These sequences are generated by means of a computer simulation of a fairly detailed kind, and this tool, together with a balance program to allocate the work to the assembly stations can solve the problem of how many stations are needed for a given mixed-model requirement. The most effective sequences are generated in conditions where assembly workers may move out of their stations by a distance equal to about one third of the length of their stations downstream and by a smaller amount upstream. Further, in conditions of uneven allocation of work the length of assembly stations must be tailored to the work allocated unless a considerable increase in line length is acceptable.

6.2 Concurrence of work

Analysis of the results of computer simulations of assembly operations led to interesting conclusions, which have been discussed in detail in the thesis. In particular the simulator results established that the most important factor affecting assembly performance is whether or not

concurrent working is permissible. A certain amount of downstream working is, in practice, unavoidable with uneven work allocations. Then, if it is important to restrict the length of the assembly line, this downstream working will take place in the next station. Here, mainly because of uneven work allocations, it will often happen that the worker in the downstream station is ready to start work. If, because concurrent working is not permissible, he must wait until the operator before him in the line has finished his task, much idle time will be involved.

In the simulation it has been assumed either that concurrent working is always possible or that it is not. There will obviously be circumstances in practical work where concurrence of work is only sometimes permissible. If matters can be arranged in such a way that across the boundaries of certain pairs of stations concurrent work may always take place and across others it may never take place, then the computer simulation described in this thesis would be sufficient after minor modifications. If however every time the question of concurrence arose all the possible combinations of tasks had to be checked through, the simulator program would be greatly increased in complexity. This is a serious disadvantage and is

discussed later in another context.

If it is assumed that the aim in industrial engineering is to produce work of an acceptable standard as cheaply as possible, then two important principles emerge from the foregoing discussion. The first is that it may be cheaper to pay for the redesign of certain assemblies to increase the content of tasks that may be done concurrently than to accept inefficient assembly operations brought about by existing inadequacy in design. The second is that the planning of operations is an item that must be paid for, and it will be worth examining the relative costs of accepting a lower standard of assembly efficiency in order to simplify the planning and hence reduce its costs.

It is considered that the matter of concurrence of work and its relations with engineering design and with the planning of assembly operations is an area for further research.

6.3. Prediction of assembly performance

The next important general matter that came out of the simulator studies was that throughout the running the performance of the assembly line varied comparatively slowly with changes in parameters. In the conditions used in the simulator runs there were no instances of

grossly non-linear performance. There is good reason, therefore, to regard an assembly line simulation as a reliable tool for predicting assembly line performance. Further, it seems probable that as knowledge is gained of the behaviour of a particular assembly line it will be possible to use the simulation to increasingly good effect.

There are two aspects to the use of a simulator in these circumstances. The first is to use it as a tool that, in the given conditions, will determine the correct number of operators to employ and that will specify a suitable sequence of models to be fed to them. However, as confidence in a simulation is built up it will be possible to use it to identify weaknesses in the given conditions, and from the results of the simulation alone to initiate action to modify these conditions.

This important function of identifying weaknesses is related to the degree of detail in which it is necessary to carry out the simulation if useful results are to be obtained. The simulator results showed, for example, that the distribution of work over stations was affected in an important way by the sequence of units. Further, it appeared that the mode of detailed behaviour would adhere

to the same general pattern over small changes, but would change significantly if major changes were made in the conditions. It is therefore important for the simulation to be sufficiently detailed to retain these patterns of behaviour. For example, exclusion from the simulation of a facility for dealing with the partly completed models of the preceding shift could easily lead to detailed errors in distribution of utility work and idle time, although the simulation might give a reasonably good representation of the overall efficiency of the assembly operation.

It follows from the arguments presented above that it is important to carry out a detailed validation of an assembly line simulation. In order to do this, it is first necessary to have an exact knowledge of the conditions of the assembly, including stochastic variations in achieved task durations, information on concurrence of work, and full details of special requirements. Next a detailed computer simulation program is required that incorporates all these data in sufficient detail. Finally, arrangements are necessary to measure the assembly-line performance in the detail needed for comparisons to be made between simulator results and actual performance. The way in which the qualified personnel chosen to undertake such work should

be organized is an important matter, which is discussed later in more detail.

The stochastic variations in the task durations may pose a difficult problem. If the variance of these tasks is small, then behaviour of the simulated assembly line may show little difference between two runs done with different samples of task duration drawn from the same population. However, if the variance in task duration is large the whole behaviour pattern of the line may alter, and detailed findings for one run may have no application to the next. This is a most important area for further research.

6.4 Speed and practicability of computations

In order that the automated planning techniques discussed above may be implemented, it is essential that the computer programs shall be fast enough to keep costs within bounds and that it shall be a practical proposition to apply them to the problem to be solved.

The following figures will serve as a general guide for speed of computation with the CDC 6400 computer as installed at the University of Adelaide. If the aggregated task group balance method is used, a single iteration of a balance allocation for 15 sixty-task models can be made in

about 0.5 seconds. This entails a balance for some 190 aggregated task groups. The ensuing simulation of an 8 hour shift for the assembly of these 15 models, using 10 stations, takes about 3 seconds. These speeds are considered well within the requirements, and it will be of value to assess the probable performance with a bigger problem.

Consider therefore an assembly line designed to cater for 20 significantly different models each containing about 600 elemental tasks. If high speed is to be obtained it is important for all the problem data to be held simultaneously in the immediate access storage of the computer. For the balance program, assuming 60-bit words and the same program organization as that given in this thesis requirements for storage would be as follows:-

	Words
Two precedence matrices (for both)	4000
Model identification vectors	2000
Elemental task duration) Hold within spare space in	
Positional weights) words of precedence matrices	
Task group durations	2000
Other variables	2000
Program and computer system requirements (say)	10000
Total	20000

The total storage required is about 20,000 words which is well within the capacity of typical medium/large computer

systems. For the simulations, the requirements for storage would be less onerous that for the balance program. Computer storage requirements therefore would not present a serious problem, provided that the bit-by-bit packing techniques discussed in the thesis were used.*

The design of the balance program is such that the time required for computation should be in a roughly linear relation to the number of aggregated task groups. It would therefore be reasonable to expect the balance to take some 5 seconds. For the simulation, provided that the number of models is about the same, a similar linear relation would apply. Double the number of models would probably increase the computational time by a factor of nearly 2. A single run for the 600 task assembly line could therefore be expected to take between 30 seconds and 1 minute. Provided that this one run was sufficient (as it could be if task duration variance was small) the costs of the computation would not be heavy. However, if task duration variance was large there would be a very real problem of computing costs to be solved. The answer to this

---

* It is worth noting that the bit-by-bit packaging techniques recommended here have a direct application to the computer solution of critical path network problems, particularly when they are characterised by activity-on-node networks.

problem will not come from a single validation operation. Research is required into assembly task duration variance and its effects in different industries, and into methods for reducing this variance and its effects.

6.5 Organization for planning of assembly operations

In order to carry out the work required for this thesis a set of computer programs containing some 3500 FORTRAN statements (excluding comments statements) has been written. This has entailed the use of some 2 hours of central processor time and 45 hours of peripheral processor time on the CDC 6400 computer and some 700 returns to the computer in the development of the programs. Despite this considerable amount of work, the programs cover only a limited number of the options possible in a limited assembly environment. A full scale package, if it were a possibility, that covered every possible variation in assembly techniques and in special assembly requirements would be so large and cumbersome that it would be impracticable to use it. One obvious alternative would be for individual industrial organizations to attempt to develop their own software. The difficulty of this approach is that it would tend to recreate in miniature some of the disadvantages of the large package discussed above, as follows. In general, it

can be expected that the programming staff employed on software development of this kind would be skilled in computing but not familiar with the details of the assembly operations involved. As a result there would again be a tendency to develop a program package of wide application. The most probable outcome of this approach would be a complicated program that would be difficult and costly to run, but would yet fail to cater for all the optional requirements of the various assembly operations. The usual fate of such programs is to be left unused.

If a single person was available who had the double qualification that he was familiar with and fully understood the engineering processes of some assembly line and if he was also a skilled computer programmer then the problem could be solved. It would be a comparatively simple matter for such a person to take basic balance and simulation programs and develop them to suit the exact requirements of a given assembly operation. In short, the requirement is for a trained and experienced industrial engineer who also is trained and experienced in the application of computers to assembly problems.

It also appears that the control of such personnel would be a matter for senior men in the industrial engineer-

ing part of an organization and not in its computing department. It is concluded that there is a real and urgent need for competent industrial engineers to be trained in the techniques of applying computers to industrial problems in general and to assembly operations in particular.

In conclusion, it seems probable that industrial processes will become more complicated and that the costs of computer hire will fall. This is an environment in which the use of computers in planning and control of industrial operations will become increasingly attractive. It appears important that properly trained people should be available to exploit this opportunity.

APPENDIX I

Table 1. Allocations of work to stations by models, simulation input 1

Model	Station number								
	1	2	3	4	5	6	7	8	9
1	2.25	2.05	1.85	2.25	2.05	1.85	2.25	2.05	1.85
2	2.15	2.55	2.35	2.15	2.55	2.35	2.15	2.55	2.35
3	2.50	2.30	2.70	2.50	2.30	2.70	2.50	2.30	2.70
4	2.85	2.65	2.45	2.85	2.65	2.45	2.85	2.65	2.45
5	2.60	3.00	2.80	2.60	3.00	2.80	2.60	3.00	2.80
6	2.95	2.75	3.15	2.95	2.75	3.15	2.95	2.75	3.15
7	3.45	3.25	3.05	3.45	3.25	3.05	3.45	3.25	3.05

Table 2. Allocations of work to stations by models, simulation input 2

Model	Station number								
	1	2	3	4	5	6	7	8	9
1	3.43	2.63	0.00	0.08	1.41	2.41	0.93	2.97	2.80
2	2.33	2.46	2.46	1.88	2.52	1.69	2.55	2.97	1.98
3	2.33	1.82	2.29	3.17	2.52	2.37	3.76	2.07	1.98
4	2.33	1.99	3.36	2.88	2.81	2.37	3.32	2.07	1.98
5	1.11	2.80	4.34	4.04	3.09	3.09	4.22	1.45	1.85
6	2.08	3.14	4.17	4.04	3.41	3.07	3.00	2.35	2.80
7	3.24	3.14	4.17	4.04	3.41	3.23	2.38	3.38	2.39

Table 3. Simulator test 1, no sequencing constraint

Simulation input 1 used throughout: a=5, d=20, throughout.

Run	Con-current work	Test input number	b	c	Up-stream allow-ance (min.)	Down-stream allow-ance (min.)	Results (man minutes)				Remarks
							Work defic-iency	Idle time	Utility work	Con-gestion	
01	Yes	1	1	2	0.50	1.50	104.14	29.25	0.00	5.30	No sequencing
02	No	1	1	2	0.50	1.50	87.52	31.29	0.00	6.33	applied
1	Yes	1	-1	0	0.50	1.50	33.79	5.54	0.00	222.42	Sequenced
2	Yes	1	-1	0	0.50	1.25	38.73	6.29	0.10	177.54	"
3	Yes	1	-1	0	0.50	1.00	41.36	6.27	0.02	129.22	"
4	Yes	1	-1	0	0.25	1.75	14.29	6.28	0.00	263.28	"
5	Yes	1	-1	0	0.25	1.50	18.27	7.54	0.00	203.14	"
6	Yes	1	-1	0	0.25	1.25	19.50	6.98	0.00	145.96	"
7	Yes	1	1	2	0.50	1.50	40.85	7.03	0.00	167.37	"
8	Yes	1	1	2	0.50	1.25	38.92	7.03	0.00	144.50	"
9	Yes	1	1	2	0.50	1.00	39.87	7.13	0.00	103.91	"
10	Yes	1	1	2	0.25	1.75	24.10	8.53	0.00	217.89	"
11	Yes	1	1	2	0.25	1.50	20.88	8.53	0.00	198.18	"
12	Yes	1	1	2	0.25	1.25	21.84	8.53	0.00	148.17	"
13	No	1	1	2	0.50	1.50	27.60	31.76	0.00	132.12	"
14	No	1	1	2	0.50	1.25	28.24	31.36	0.00	89.30	"
15	No	1	1	2	0.50	1.00	28.24	31.16	0.00	70.15	"
16	No	1	1	2	0.25	1.75	12.83	32.72	0.00	199.57	"
17	No	1	1	2	0.25	1.50	12.83	33.12	0.00	139.25	"
18	No	1	1	2	0.25	1.75	12.83	32.92	0.00	124.65	"

Station passage time 3.45 min. (same for each station)
Shift duration 158.12 min.
Effort supplied 1423.08 min.
Launch interval 2.68 min.
Mix by models 10(1), 5(2), 15(3), 7(4), 6(5), 7(6), 9(7)
+ uncompleted units from preceding shift.

Table 4. Simulator Test 1, Sequences

Sequence 1. Hand prepared, evenly distributed sequence used as input for runs 01, 02, Table 3

Position of units in the sequence	Models in the sequence									
1 to 10	4	5	5	4	3	4	5	5	4	3
11 to 20	4	5	3	1	2	3	4	5	7	1
21 to 30	3	6	5	4	3	1	2	7	3	5
31 to 40	4	1	3	7	6	2	3	1	6	4
41 to 50	3	7	5	1	3	7	6	4	3	1
51 to 60	2	5	3	7	6	1	3	4	7	6
t1 to 70	3	1	2	4	3	5	7	1	3	7
71 to 80	6	4	3	7	4	3	2	1	5	6

Sequence 2. Sequence achieved by sequencing algorithm 2, run 18, Table 3

Position of units in the sequence	Models in the sequence									
1 to 10	4	5	5	4	3	4	5	5	4	3
11 to 20	4	5	6	1	7	1	6	1	7	1
21 to 30	6	1	6	1	7	1	6	1	6	1
31 to 40	7	1	6	2	5	2	5	2	5	2
41 to 50	7	3	7	2	7	3	5	3	7	3
51 to 60	4	3	5	3	5	3	4	3	4	3
61 to 70	4	3	4	3	4	3	7	3	4	3
71 to 80	3	-	-	-	-	-	-	-	-	-

APPENDIX I

Table 5. Simulator test 2, no sequencing constraint

Simulation input 2 used throughout: a=5, b=1, c=2, d=20 in all runs

Run	Concurrent work	Test input number	Upstream allowance (min.)	Downstream allowance (min.)	Results (man minutes)				Remarks
					Work deficiency	Idle time	Utility work	Congestion	
01	Yes	2	0.50	1.50	54.08	27.32	6.22	92.52	Not sequenced
02	No	2	0.50	1.50	17.86	36.40	10.07	126.55	Sequenced
1	Yes	2	0.50	1.50	55.60	25.62	0.41	94.23	"
2	Yes	2	0.50	1.25	58.69	28.27	0.84	74.34	"
3	Yes	2	0.50	1.00	68.84	41.91	0.82	38.15	"
4	Yes	2	0.25	1.75	25.19	29.69	1.15	159.72	"
5	Yes	2	0.25	1.50	23.40	28.83	0.41	112.90	"
6	Yes	2	0.25	1.25	26.82	36.82	0.44	68.68	"
7	No	2	0.50	1.50	23.58	42.39	2.56	109.38	"
8	No	2	0.50	1.25	26.51	50.23	1.54	75.14	"
9	No	2	0.50	1.00	37.01	56.49	3.92	44.73	"
10	No	2	0.25	1.75	15.22	55.50	1.01	88.48	"
11	No	2	0.25	1.50	7.28	46.09	4.90	121.43	"
12	No	2	0.25	1.25	10.88	48.76	3.94	86.90	"

Station passage time	4.34 min.
Shift duration	161.20 min.
Effort supplied	1450.80 min.
Launch interval	2.60 min.
Mix by models	15(1), 8(2), 11(3), 4(4), 5(5), 9(6), 10(7) + uncompleted units from preceding shift

APPENDIX I

Table 6. Simulator Test 2, Sequences

Sequence 1. Hand prepared, evenly distributed sequence used as input to runs 01, 02, Table 5 Appx. I

Position of units in the sequence	Models in the sequence									
1 to 10	7	2	1	5	4	1	7	2	1	5
11 to 20	4	1	6	7	2	1	~~7~~3	~~8~~4	~~4~~5	~~2~~1
21 to 30	3	6	7	1	3	7	6	1	3	4
31 to 40	5	1	3	6	7	1	3	6	5	1
41 to 50	7	2	3	1	6	7	2	1	2	4
51 to 60	5	1	3	6	7	1	2	6	7	1
61 to 70	2	7	6	1	3	7	6	1	5	4
71 to 80	2	1	7	2	3	1				

Sequence 2. Sequence achieved by sequencing algorithm 2, run 10, Table 5, Appx. I

Position of units in the sequence	Models in the sequence									
1 to 10	7	2	1	5	4	1	7	2	1	5
11 to 20	4	1	6	7	2	1	7	3	4	2
21 to 30	4	2	6	1	7	3	4	1	6	2
31 to 40	6	1	7	3	4	3	2	1	6	2
41 to 50	3	1	7	3	6	1	6	2	6	1
51 to 60	5	1	3	2	5	1	6	1	3	2
61 to 70	3	1	7	3	3	1	6	1	7	1
71 to 80	5	1	7	5	7	5	7			

Table 7. Simulator test 3, sequencing under constraint a=5, b=1, c=2, d=20 in all runs

Run	Concurrent work	Test input number	Stack size	Look ahead	Results (man minutes)				Remarks
					Work deficiency	Idle time	Utility work	Congestion	
1	No	1	4	5	70.31	32.75	0.00	9.86	
2	No	1	6	5	56.85	30.05	0.00	10.92	
3	No	1	8	5	75.60	31.95	0.00	2.87	All runs 1 to 21 were automatically sequenced
4	No	1	4	8	60.11	30.48	0.00	24.81	
5	No	1	6	8	54.03	23.54	0.00	8.51	
6	No	1	8	8	54.43	23.62	0.00	16.14	
7	No	1	4	16	42.12	28.55	0.00	32.79	
8	No	1	6	16	36.94	21.43	0.00	16.01	
9	No	1	8	16	38.76	19.82	0.00	23.26	
10	Yes	2	4	5	48.60	25.33	10.79	78.31	
11	Yes	2	6	5	41.00	21.00	6.96	72.08	
12	Yes	2	8	5	38.23	20.98	8.43	89.29	
13	Yes	2	4	16	44.62	32.04	19.23	91.99	
14	Yes	2	6	16	42.32	24.83	12.89	100.06	
15	Yes	2	8	16	42.94	26.68	19.83	131.52	
16	No	2	4	5	30.46	39.79	15.48	87.24	
17	No	2	6	5	27.92	40.59	12.34	84.19	
18	No	2	8	5	36.04	44.44	13.31	67.15	
19	No	2	4	16	28.98	49.31	25.20	119.71	
20	No	2	6	16	33.21	40.25	19.25	93.34	
21	No	2	8	16	30.12	36.08	18.14	111.37	

(Upstream allowance = 0.50 for all runs (Downstream allowance = 1.25

(continued)

Table 7. Simulator test 3, sequencing under constraint (contd.)

a=5, b=1, c=2, d=20 in all runs

Run	Con-current work	Test input number	Stack size	Look ahead	Results (man minutes)				Remarks
					Work defic-iency	Idle time	Utility work	Con-gestion	
01	Yes	1	-	-	97.92	31.56	0.00	39.62	Not sequenced
02	No	1	-	-	70.68	34.69	0.00	64.83	Upstream allowance = 0.50
03	Yes	2	-	-	53.47	55.81	34.70	136.49	
04	No	2	-	-	23.35	68.33	42.93	191.21	Downstream allowance = 1.50

Station passage time	3.45 min. (input 1) and 4.34 min. (input 2)
Shift duration	158.12 (input 1) and 161.20 (input 2)
Effort supplied	1423.08 man min. (input 1) and 1450.80 min. (input 2)
Launch interval	2.68 min. (input 1) and 2.60 min. (input 2)

Table 8. Simulator Test 3, Sequences

Sequence I. Hand-prepared sequence bunched in groups of high and low work-content models

Position of units in the sequence	Models in the sequence									
1 to 10	4	5	5	4	3	4	5	5	4	3
11 to 20	4	5	3	1	2	3	4	5	7	1
21 to 30	3	6	5	7	6	1	2	4	3	2
31 to 40	4	1	3	3	3	5	7	1	6	4
41 to 50	7	7	5	1	3	3	6	7	6	1
51 to 60	2	3	3	4	3	1	5	4	7	6
61 to 70	3	1	2	4	3	1	7	5	3	7
71 to 80	6	7	3	4	4	3	2	1	5	6
81 to 90	7	6	4	5	6	2	3	4	1	1
91 to 100	3	3	7	5	3	6	5	1	7	2

Number of uncompleted products from preceding shift is 12. The greatest number of products on which work was started in any run of table 6 was 77, giving, for a look ahead of 16, a maximum requirement in the sequence of 93 product units.

Sequence 2. Sequence achieved by sequencing algorithm 4, run 9, Table 7, Appx. I

Position of units in the sequence	Models in the sequence									
1 to 10	4	5	5	4	3	4	5	5	4	3
11 to 20	4	5	6	3	7	1	7	1	6	2
21 to 30	5	3	7	4	2	6	1	3	5	4
31 to 40	5	3	6	2	7	1	6	4	3	5
41 to 50	3	3	1	5	4	7	1	6	3	4
51 to 60	3	4	1	5	7	2	7	3	7	3
61 to 70	2	3	1	7	3	7	1	7	1	6

APPENDIX II

Table 1. Elemental task durations

Elemental Task Number	Elemental Task Duration									
1 to 10	0.46	0.29	1.10	0.21	0.16	0.65	0.48	0.30	0.10	0.47
11 to 20	0.31	0.47	0.31	0.26	0.49	0.21	0.64	0.33	0.41	0.28
21 to 30	0.42	0.28	0.94	0.78	0.54	0.22	0.22	0.38	0.36	0.17
31 to 40	0.33	0.74	0.21	0.17	0.17	0.09	0.09	0.36	0.79	0.24
41 to 50	0.81	0.81	0.40	0.53	0.17	0.12	0.08	0.57	0.57	0.44
51 to 60	0.86	0.68	0.81	0.22	0.71	0.71	0.89	0.06	0.46	0.60
61 to 70	0.30	0.36	0.45	0.44	0.40	0.29	0.51	0.48	1.10	0.96
71 to 80	0.48	0.96	0.47	0.54	0.43	0.48	0.47	0.73	0.60	1.38
81 to 90	0.38	0.26	0.37	0.87	0.87	0.26	0.37	0.76	0.35	0.66
91 to 100	0.24	1.11	1.04	1.03	0.32	0.63	0.17	0.55	0.00	0.80
101 to 110	0.39	0.80	0.39	0.30	0.46	0.18	0.30	0.28	0.00	0.00
111 to 120	0.52	1.48	0.45	0.74	0.28	0.67	0.26	0.48	0.38	0.48
121 to 130	0.34	0.00	1.18	1.18	1.50	1.50	0.44	0.88	1.02	1.84
131 to 140	1.54	1.28	2.16	0.24	0.28	0.00	0.00	0.39	0.67	0.56
141 to 150	0.43	0.95	0.11	0.08	1.16	0.36	1.08	0.47	0.76	0.12
151 to 160	0.00	0.55	0.55	0.19	0.23	0.11	0.09	0.14	0.15	0.15
161 to 170	0.36	0.20	0.16	0.16	0.32	0.15	0.07	0.00	0.00	0.58
171 to 180	0.65	0.70	1.93	0.22	0.22	1.12	0.29	0.11	0.00	0.00
181 to 190	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
191 to 200	0.00									

Tasks 179 to 191 have been set to zero to exclude them from effective part in the balance. They are specialist tasks not undertaken by the assembly line operators.

Table 2. Examples of model identification vectors

Task number	Model number														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
94	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
95	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
96	0	0	0	0	0	1	0	1	0	0	1	1	1	1	0
97	0	0	0	0	0	1	0	1	0	0	1	1	1	1	0
98	0	0	0	0	1	1	1	1	1	1	1	1	1	1	0

APPENDIX II

Table 3. Model work content and requirements, application with first model mix

Model number	Work content (man min.)	Numbers initially required	Requirement as adjusted by balance program
1	22.51	4	3
2	22.51	4	3
3	22.51	2	1
4	22.51	6	5
5	25.15	8	7
6	26.07	17	16
7	26.13	2	2
8	27.05	2	2
9	28.89	56	56
10	29.87	1	1
11	29.96	28	28
12	30.64	10	10
13	30.94	1	1
14	31.62	2	2
15	43.93	14	14

Table 4. Balance achieved, application with first model mix

Station number	Total of task groups allocated	Station residue
1	444.96	5.04
2	448.30	1.70
3	448.57	1.43
4	445.59	4.41
5	449.95	0.05
6	448.90	1.10
7	445.74	4.26
8	448.75	1.25
9	448.80	1.20
10	442.28	7.72
Totals	4471.84	28.16

APPENDIX II

Table 5. Example of task group splitting

Station	Task group number	Position of task group in station	Magnitude of task group (man min.)	Running total of allocated task groups (man min.)
2	25	11	75.06	429.77
	13	12	3.72	433.49
	15	13	5.88	439.37
	14	14	3.12	442.49
	16	15	2.52	445.01
	12	16	3.29	448.30
3	12	1	62.04	62.04
	35	2	23.63	85.67
	⋮	⋮	⋮	⋮

Table 6. Allocations of work to stations by models, application with first model mix

Model number	Station number 1	2	3	4	5	6	7	8	9	10
1	0.94	2.15	5.12	2.06	4.26	3.27	0.0	2.51	0.44	1.76
2	0.94	2.15	5.12	2.06	4.26	3.27	0.0	2.51	0.44	1.76
3	0.94	2.15	5.12	2.06	4.26	3.27	0.0	2.51	0.44	1.76
4	0.94	2.15	5.12	2.06	4.26	3.27	0.0	2.51	0.44	1.76
5	3.12	3.54	2.31	2.39	2.21	2.23	2.96	1.16	3.08	2.15
6	3.12	3.07	2.78	1.95	3.47	2.71	2.58	1.16	3.08	2.15
7	3.12	2.13	3.56	3.53	2.21	2.23	2.96	1.16	3.08	2.15
8	3.12	2.13	2.78	3.09	3.47	2.71	2.58	1.16	3.08	2.15
9	3.12	3.07	2.78	2.90	2.21	1.78	3.14	4.17	3.10	2.62
10	3.12	2.13	3.56	4.04	2.21	1.78	3.14	4.17	3.10	2.62
11	3.12	3.07	2.78	2.40	3.83	2.71	3.96	1.16	3.66	3.27
12	3.12	3.07	2.78	2.46	3.47	2.71	3.14	4.17	3.10	2.62
13	3.12	2.13	3.56	3.54	3.83	2.71	3.96	1.16	3.66	3.27
14	3.12	2.13	3.56	3.60	3.47	2.71	3.14	4.17	3.10	2.62
15	3.12	3.07	2.62	6.48	2.70	9.16	2.94	4.71	2.88	6.25
Station passage time (min.)	4.00	4.00	5.00	6.50	5.00	9.00	4.00	5.00	4.00	6.50

Table 7. Application with first model mix, Sequences

Sequence 1. Hand prepared evenly distributed sequence

Position of unit in sequence	Models in the sequence				
1 to 20	6 9 11 12 9	15 1 9 11 6	9 5 12 9 11	7 9 15 4 9
21 to 40	11 9 11 6 9	5 15 9 9 11	9 1 6 9 11	15 9 4 13 9
41 to 60	11 6 9 2 15	9 11 7 9 5	12 9 11 4 9	3 6 9 11 9
61 to 80	9 11 6 9 11	15 9 5 4 9	11 6 9 15 1	9 11 4 9 14
81 to 100	12 9 11 6 9	15 9 9 11 5	9 12 6 9 11	9 9 2 15 9
101 to 120	11 8 9 6 15	9 11 5 9 9	12 9 8 11 9	15 12 9 11 6
121 to 140	9 15 6 9 11	1 9 10 6 9	11 15 9 5 7	9 11 6 9 11
141 to 160	15 9 11 6 9	2 5 9 11 12	9 15 6 9 11	12 9 12 12 9
161 to 180	11 14 9 12 6	9 11 15 9 12	5 9 11 4 9	12 6 9 6 9
181 to 200	11 6 9 5 15	9 9 11 9 1	6 9 11 15 9	4 13 9 11 6

Sequence 2. Sequence generated by sequencing algorithm 1, run 1 Table 6

1 to 20	6 9 11 12 9	15 1 9 11 6	9 5 12 9 11	7 9 15 8 12
21 to 40	14 6 14 5 12	6 4 9 13 9	15 5 12 5 12	8_{10} 4_{11} 9 13_{12} 8_1
41 to 60	8 9 12 5 15	5 12 7 12 5	12 9 12 11 15	7 11 1 9 9
61 to 80	11 9 11 9 15	6 9 6 9 1	11 9 11 9 9	5 15 6 11 2
81 to 100	9 9 11 9 11	9 15 6 9 6	11 9 6 9 4	9 15 11 11 9
101 to 120	9 6 11 9 2	9 9 11 15 6	9 6 9 9 11	9 9 6 15 6
121 to 140	9 2 11 9 11	9 9 15 6 6	11 9 4 9 11	9 11 9 15 6
141 to 160	9 9 4 11 9	9 11 9 ~~15~~4 8_4	11 15 3 9 9	11 9 9 15 11
161 to 180	11 9 11 9 15	9 9 9 9 --	- - - - -	- - - - -
181 to 200				

Table 7. Application with first model mix, Sequences (contd.)

Sequence 3. Final models of sequence of run 10, Table 8 Appx. II

Position	Models in the sequence																			
141 to 160	11	9	11	9	11	9	9	4	9	9	9	9	9	4	9	9	9	9	9	9
161 to 169	9	9	9	9	9	9	9	9	9	-										

Sequence 4. Final models of sequence of run 11, Table 8 Appx. II

Position	Models in the sequence																			
141 to 160	9	3	9	9	9	9	9	4	9	9	9	9	9	4	9	9	9	9	9	9
161 to 169	9	9	9	9	9	9	4	4	4	-										

Sequence 5. Final models of sequence of run 18, Table 8 Appx. II

Position	Models in the sequence																			
141 to 160	9	12	6	9	12	15	9	11	6	9	14	5	9	11	6	9	15	12	9	11
161 to 169	4	9	12	9	15	11	9	12	9	-										

Table 8. Simulation results, application with first model mix

Run	Concurrent work	Upstream allowance (min.)	Downstream allowance (min.)	Stack size	Look ahead	Performance parameters (man min.)				Remarks
						Work deficiency	Idle time	Utility work	Congestion	
1	Yes	0.50	1.50	-	-	89.47	54.56	3.80	281.20	
2	Yes	0.50	1.25	-	-	92.38	59.36	5.80	284.91	
3	Yes	0.50	1.00	-	-	114.18	82.40	22.98	252.12	
4	Yes	0.25	1.75	-	-	42.15	63.36	5.41	400.55	
5	Yes	0.25	1.50	-	-	38.63	50.50	3.20	351.63	
6	Yes	0.25	1.25	-	-	47.98	73.03	11.95	326.18	Unconstrained sequencing
7	No	0.50	1.50	-	-	47.28	140.11	120.31	577.85	
8	No	0.50	1.25	-	-	56.97	148.70	130.78	477.28	
9	No	0.50	1.00	-	-	58.19	190.62	196.21	508.05	
10	No	0.25	1.75	-	-	19.68	149.89	127.29	818.87	
11	No	0.25	1.50	-	-	20.71	169.48	163.69	816.59	
12	No	0.25	1.25	-	-	24.91	144.97	126.45	471.31	
13	Yes	0.50	1.25	6	5	111.38	71.94	26.55	225.61	Sequencing under constraint
14	Yes	0.50	1.25	6	8	117.28	77.29	30.13	247.85	
15	Yes	0.50	1.25	6	16	110.00	75.73	32.09	231.45	
16	No	0.50	1.25	6	5	49.96	108.54	52.21	415.77	
17	No	0.50	1.25	6	8	60.11	116.30	64.70	392.32	
18	No	0.50	1.25	6	16	52.21	118.32	64.71	407.66	
19	Yes	0.50	1.50	-	-	113.46	83.75	39.63	369.06	Not sequenced

Shift duration 450.00 min.
Effort supplied 4500.00 man min.
Launch interval 2.98 min.
Mix by models 3(1), 3(2), 1(3), 5(4), 7(5), 16(6), 2(7), 2(8), 56(9), 1(10), 28(11), 10(12), 1(13), 2(14), 14(15).

Table 9. Distribution of idle time and utility work over stations, application with first model mix

Run number corresponding to runs in Table 8 Appx. II	Idle time (man min.)			Utility work (man min.)			Remarks
	Station number			Station number			
	6	7	8	6	7	8	
1	1.06	8.42	11.09	0.00	1.60	1.75	Concurrent work permissible. Unconstrained sequencing
2	1.13	10.08	9.99	0.00	0.84	2.64	
3	5.03	16.44	10.15	0.00	5.32	8.53	
4	1.95	10.59	10.37	0.00	0.36	2.61	
5	0.60	9.06	5.86	0.00	0.05	1.65	
6	4.35	13.03	10.16	0.00	2.48	5.28	
7	24.58	21.40	29.62	38.58	18.96	10.95	Concurrent work not permissible. Unconstrained sequencing
8	25.57	21.88	32.22	39.57	19.48	14.08	
9	46.06	14.63	40.32	54.10	25.11	29.79	
10	25.13	24.07	33.35	40.37	18.60	13.19	
11	38.88	19.12	36.36	49.65	23.07	21.51	
12	24.31	22.52	30.87	38.06	19.55	12.73	
13	2.25	11.74	15.20	1.54	3.78	7.29	Concurrent work permissible. Sequencing under constraint.
14	3.41	11.91	16.84	3.42	3.63	5.82	
15	0.50	15.56	18.36	1.10	7.44	10.45	
16	4.03	16.05	29.70	2.63	7.70	17.85	Concurrent work not permissible. Sequencing under constraint.
17	8.01	16.21	30.97	8.02	8.68	18.79	
18	6.73	17.08	32.70	6.74	7.59	23.97	
19	12.62	17.01	9.60	4.52	8.76	4.60	Hand prepared sequence.

Table 10. Model work content and requirements, application with second model mix

Model number	Work content (man min.)	Numbers initially required	Requirement as adjusted by balance program
1	22.51	14	13
2	22.51	4	3
3	22.51	2	1
4	22.51	16	15
5	25.15	8	7
6	26.07	17	16
7	26.13	2	1
8	27.05	2	1
9	28.89	56	55
10	29.87	1	0
11	29.96	28	27
12	30.64	10	10
13	30.94	1	1
14	31.62	2	2
15	43.93	14	14

Table 11. Balance achieved, application with second model mix

Station number	Total of task groups allocated	Station residue
1	477.10	2.90
2	477.57	2.43
3	478.74	1.26
4	479.21	0.79
5	478.73	1.27
6	475.96	4.04
7	476.72	3.28
8	475.07	4.93
9	474.99	5.01
10	455.41	24.59
Totals	4749.50	50.50

Table 12. Allocations of work to stations by models, application with second model mix

Model number	Station number									
	1	2	3	4	5	6	7	8	9	10
1	1.82	1.27	5.12	3.03	3.29	3.09	1.03	2.10	0.88	0.88
2	1.82	1.27	5.12	3.03	3.29	3.09	1.03	2.10	0.88	0.88
3	1.82	1.27	5.12	3.03	3.29	3.09	1.03	2.10	0.88	0.88
4	1.82	1.27	5.12	3.03	3.29	3.09	1.03	2.10	0.88	0.88
5	3.12	3.33	2.52	2.39	2.21	2.61	2.58	1.39	2.85	2.15
6	3.12	3.33	2.52	1.95	2.84	3.34	2.58	1.39	2.85	2.15
7	3.90	2.39	3.20	2.85	2.21	2.61	2.58	1.39	2.85	2.15
8	3.90	2.39	3.20	2.41	2.84	3.34	2.58	1.39	2.85	2.15
9	3.12	3.33	2.52	2.90	2.21	1.78	3.57	3.74	3.02	2.70
10	3.90	2.39	3.20	3.36	2.21	1.78	3.57	3.74	3.02	2.70
11	3.12	3.33	2.52	1.95	3.65	3.34	3.96	1.39	2.85	3.85
12	3.12	3.33	2.52	2.46	2.84	3.34	3.57	3.74	3.02	2.70
13	3.90	2.39	3.20	2.41	3.65	3.34	3.96	1.39	2.85	3.85
14	3.90	2.39	3.20	2.92	2.84	3.34	3.57	3.74	3.02	2.70
15	3.12	3.24	0.83	6.18	3.75	5.09	2.34	6.24	6.89	6.25
Station passage time (min.)	4.00	3.50	5.00	6.00	4.00	5.00	4.00	6.00	7.00	6.00

Table 13. Simulation results, application with second model mix

Run	Concurrent work	Upstream allowance (min.)	Downstream allowance (min.)	Stack size	Look ahead	Performance parameters (man min.)				Remarks
						Work deficiency	Idle time	Utility work	Congestion	
1	Yes	0.50	1.50	-	-	114.68	82.77	8.98	411.30	Unconstrained sequencing
2	Yes	0.50	1.25	-	-	133.62	97.94	11.57	264.31	
3	Yes	0.50	1.00	-	-	144.16	101.05	15.35	194.39	
4	Yes	0.25	1.75	-	-	47.08	81.94	8.07	516.06	
5	Yes	0.25	1.50	-	-	50.60	83.25	7.28	423.17	
6	Yes	0.25	1.25	-	-	59.19	101.84	14.85	283.86	
7	No	0.50	1.50	-	-	45.63	153.15	102.71	641.30	
8	No	0.50	1.25	-	-	64.82	133.17	88.46	467.02	
9	No	0.50	1.00	-	-	81.89	161.57	118.75	350.63	
10	No	0.25	1.75	-	-	25.46	180.48	122.86	816.05	
11	No	0.25	1.50	-	-	24.33	159.88	110.95	647.17	
12	No	0.25	1.25	-	-	33.39	162.65	118.03	497.68	
13	Yes	0.50	1.25	6	8	92.33	71.18	37.31	343.63	Sequencing under constraint
14	Yes	0.50	1.25	6	16	107.01	84.48	47.67	377.69	
15	No	0.50	1.25	6	8	48.02	136.49	90.83	534.58	
16	No	0.50	1.25	6	16	56.08	145.13	99.73	558.81	
17	Yes	0.50	1.50	-	-	104.46	97.51	61.00	486.46	Not sequenced

Shift duration 480.00 min.
Effort supplied 4800.00 man min.
Launch interval 2.91 min.
Mix by models 13(1), 3(2), 1(3), 15(4), 7(5), 16(6), 1(7), 1(8), 55(9), 0(10), 27(11), 10(12), 1(13), 2(14), 14(15).

APPENDIX II

Table 14. Modifications to group positional weights

Task number	Former group positional weight	Positional weight after modification	Remarks
79	2438.70	700.00	Changes made to delay allocation of all two-man tasks by delaying the predecessors 79, 95, 98.
95	2362.50	700.01	
98	2107.28	700.02	
118	1884.25	1884.25	Positional weights of two-man tasks have been made larger than that of 118 to avoid possibility of task with positional weight close to that of 118 outranking one of the two-man tasks.
129	1691.13	1884.26	
132	1547.31	1884.27	
134	1366.83	1884.28	
140 to 148)	1186.89 and smaller values	1884.29 to 1884.29	
154)		1884.40	
157 to 164)		1884.41 to 1884.48	

Table 15. Balance achieved with double stations

Stations	Task group allocated	Task group magnitude	Running total of task groups allocated	Remarks
6	⋮	⋮	⋮	
	118*	3.36	478.79	Split task
7	118*	60.48	60.48	
	129*	135.66	196.14	
	132*	170.24	366.38	Precedence forces 130, 131 133 into the balance here
	130	25.76	392.14	
	131	21.56	413.70	
	133	30.24	443.94	
	134*	28.56	472.50	
	120	6.72	479.22	
8	138*	51.87	51.87	
	139*	89.11	140.98	
	140*	36.96	177.94	
	141*	28.38	206.32	
	142*	62.70	269.02	
	143*	7.26	276.28	
	144*	5.28	281.56	
	145*	154.28	435.84	
	157*	4.77	440.61	
	158*	7.42	448.03	
	159*	7.95	455.98	
	160*	7.95	463.93	
	121	4.76	468.69	
9	161*	19.08	19.08	
	162*	10.60	29.68	
	163*	8.48	38.16	
	164*	8.48	46.64	
	146*	23.76	70.40	
	147*	71.28	141.68	
	148*	31.02	172.70	
	154*	12.54	185.24	Precedence constraints cause 154 to follow 146, 147 148 despite its larger position weight
	⋮	⋮	⋮	

* Two-man task

BIBLIOGRAPHY

1. Arcus, A.L., "Comsoal, A Computer Method of Sequencing Operations for Assembly Lines" International Journal of Production Research, Vol. 4, No. 4, 1966.

2. Buffa, E.S., "Production-Inventory Systems, Planning and Control", pp 256 to 261, Richard D. Irwin Inc., 1968.

3. Held, M., Karp, R.M. and Shareshian, R., "Assembly-Line Balancing with Precedence Constraints", Journal of the Operations Research Society of America, Vol. 11, May-June 1963.

4. Helgeson, W.B. and Birnie, D.P., "Assembly-Line Balancing using the Ranked Positional Weight Technique", Journal of Industrial Engineering, Vol. XII, No. 6, Nov.-Dec. 1961.

5. Heskiaoff, H. and Weinstein, J., "A Heuristic Method of Assembly-Line Balancing" ORSA/TIMS Joint Meeting, San Francisco, California, May 1968.

6. Ignall, E.J., "A Review of Assembly-Line Balancing", Journal of Industrial Engineering, Vol. XVI, No. 2, March-April 1965.

7. Jackson, J.R., "A Computing Procedure for a Line Balancing Problem", Management Science, Vol. 2, No. 3, April 1956.

8. Kaufmann, A., "Graphs, Dynamic Programming and Finite Games", Academic Press, 1967 pp 229 to 329.

9. Kilbridge, M. and Wester, L., "A Heuristic Method of Assembly-Line Balancing", Journal of Industrial Engineering, Vol. XII, No. 4, July-August 1961.

10. Kilbridge, M. and Wester, L., "The Balance Delay Problem", Management Science, Vol. 8, October 1961.

11. Kilbridge, M. and Wester, L., "A Review of Analytical Systems of Line Balancing", Journal of the Operations Research Society of America, Vol. 10, No. 5, September-October 1962.

12. Macaskill, J.L.C., "Applications of Computers to the Analysis of Mixed-Product Assembly Lines", Proceedings of Fourth Australian Computer Conference, Adelaide, South Australia, 1969.
(A copy of this reference is included with the thesis.)

13. Mansoor, E.M., "Assembly-Line Balancing - An Improvement in the Ranked Positional Weight Technique", Journal of Industrial Engineering, Vol. XVI, No. 2 March-April 1965.

14. Mansoor, E.M. and Yadin, M., "On the Problem of Assembly-Line Balancing", The Third Annual Israel Conference on Operations Research, 1969.

15. Moodie, C.L. and Young, H.H., "A Heuristic Method of Assembly-Line Balancing for Assumptions of Constraint on Variable Work Element Times", Journal of Industrial Engineering, Vol. XVI, No. 1, January-February 1965.

16. Moreno, C.W., "A Technique for Simulating Transient Sequential Queues in Production Lines", Ph.D. Thesis, 1966, Purdue University, Lafayette, Indiana.

17. Reiter, R., "On Assembly-Line Balancing Problems", Journal of the Operations Research Society of America, Vol. 17, No. 4, July-August 1969.

18. Salveson, M.E., "The Assembly-Line Balancing Problem", Proceedings of the Second Symposium in Linear Programming, Washington, D.C., January 27-29, 1955, Vol. 1.

19. Thomopoulos, N.T., "Line Balancing - Sequencing for Mixed-Model Assembly", Management Science, Vol. 14, No. 2, October 1967.

20. Thomopoulos, N.T., "A Sequencing Procedure for a Multi-Model Assembly Line", Ph.D. Thesis, Illinois Institute of Technology, Chicago, Illinois, 1966.

21. Tonge, F.M., "Assembly-Line Balancing using Probabilistic Combinations of Heuristics", Management Science, Vol. 11, No. 2, May 1965.

22. Wester, L. and Kilbridge, M., "The Assembly Line Model-Mix Sequencing Problem", Proceedings of the Third International Conference on Operations Research, Oslo, 1963.

Macaskill, J.L.C., (1969) Application of computers to the analysis of mixed-product assembly lines.
Proceedings of the Fourth Australian Computer Conference, Adelaide, South Australia, 1969, v. 1, pp. 61-67.