1-1-1974

A systematic heuristic approach to scheduling in the small job shop.

Frank K. Pfeiffer
University of Massachusetts Amherst

A SYSTEMATIC HEURISTIC APPROACH TO SCHEDULING IN THE SMALL JOB SHOP

A dissertation

by

FRANK K. PFEIFFER, JR.

Submitted to the Graduate School of the
University of Massachusetts in partial
fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

August 1974

Major Subject: Management Information Systems

A SYSTEMATIC HEURISTIC APPROACH

TO SCHEDULING IN THE SMALL JOB SHOP

A Dissertation Presented

By

FRANK K. PFEIFFER, JR.

Approved as to style and content by:

Van Court Hare Jr., Chairman

Van Court Hare, Jr., Professor of Management

Stephen R. Michael

Stephen R. Michael, Associate Professor of Management

Max S. Wortman, Jr., Professor of Management

Hugh J. Miser

Hugh J. Miser, Professor of Industrial Engineering & Operations Research

John T. Conlon

John T. Conlon, Acting Dean
School of Business Administration

ACKNOWLEDGEMENTS

The encouragement to endure the writing of this dissertation was provided primarily by two figures: Van Court Hare, Jr., Professor of Management, and John T. Conlon, Associate Dean of the Graduate School of Business. I am extremely indebted to Professor Hare, whose experience and genius, reflected in him many comments and suggestions, have contributed immensely to the significance of this thesis. In addition, Dr. Conlon provided patience and understanding during times when any other attitude might easily have led to the abandonment of my work.

Special thanks are due also to Mrs. Vesta Powers, who typed the manuscript with an excellence not befitting the quality of my scribble; to Professor John Whenman, who transcribed my drawings with superhuman precision; and, finally, to Nichols College, in whose employ I was provided with the proper frame of mind to complete my work.

ABSTRACT

This dissertation deals with the application of practical heuristics in the small job shop. The emphasis is on heuristic devices of the "rule of thumb" variety, thus negating the need to employ extensive machine-computational facilities. Early chapters provide the reader with a survey of the scheduling literature, as well as with a discussion of fundamental scheduling concepts and terminology. Some common priority rules are subsequently discussed, and their application is illustrated with reference to a typical scheduling problem. The supplementary application of several "rules of thumb" is then suggested as a device to be used in the improvement of schedules. The flowshop problem is subsequently covered in detail, and the final part of the dissertation presents a non-parametric argument to justify the use of heuristics as a workable, highly satisfactory approach to scheduling.

Table of Contents

CHAPTER I

INTRODUCTION

In 1959, Roger Sisson attempted to summarize major accomplishments in the area of job shop scheduling.[1] The study revealed that the existing research was neither grandly conceived nor far-reaching in terms of practical conclusions. At that stage in the development of the literature, most authors seemed content to specify problems, delineate complexities, and suggest the need for further research. Some went so far as to set up and solve (or partially solve) trivial examples, but, for the most part, it was patently clear that much remained to be done.

In 1971, some twelve years later, the field was again surveyed, this time by Bravoco.[2] Astonishingly, Bravoco drew much of the same conclusion as Sisson: that the job shop scheduling problem was still very much unsolved. (Indeed, the one-machine sequencing case was still being analyzed!)

Definition of Scheduling

Since the term "scheduling" has received various interpretations when used in a job shop context, Webster's basic definition is cited as a benchmark: A schedule (def. 3) is "a usually written plan or proposal for future procedure, typically indicating the objective proposed, the time and sequence of each operation, and the materials required."[3]

In the literature, the definitions of "scheduling" and "sequencing" have frequently been confused. For our purposes, "scheduling" will be used to describe the establishment of a "program" in which sequencing (ordering) of operations is only part of the programming process. In such a program, the operations of each job are assigned to machines for processing at specified times. Operations may then be moved backwards or foreward in time (i.e. have their schedules "adjusted"), but the order of operations is not necessarily disturbed. The emphasis is therefore on the placement of operations within a time continuum, in which case any repositioning does not necessarily constitute a reordering. (In situations that deal specifically with the ordering process, the term "sequencing" will be applied.)

Objectives of the Scheduling Process

The general goal of scheduling analysis is to create schedules that are efficient in terms of some criterion or set of criteria. Typically, the approach has been geared to optimization, even though the practicality of such orientation has always been open to serious question.

The following is a representative list of commonly pursued objectives:

(1) Minimization of total processing time over a set of

jobs (minimization of "makespan").

(2) Minimization of the processing time of each job.

(3) Minimization of in-process inventory costs.

(4) Minimization of machine utilization.

(5) Minimization of the number of late jobs.

(6) Minimization of total tardiness.

(7) Minimization of costs due to not meeting due date exactly. (Relates to items 5 and 6.)

Constraints on the Scheduling Models

Even the most "general" scheduling model frequently calls for the imposition of a lengthy set of constraints, the relaxing of which renders the model invalid. The constraints are not, of course, exactly uniform from model to model, but there are, nevertheless, several which are encountered with remarkable regularity. In general, unless the model analyst specifies otherwise, most or all of the following are assumed to be binding:[4]

(1) No machine may process more than one operation at a time.

(2) Each operation, once started, must be performed to completion (no preemptive priorities).

(3) Each job, once started, must be performed to completion (no order cancellations).

(4) Each job is an entity; that is, even though the job represents a lot of individual parts, no lot may be pro-

cessed by more than one machine at a time. This condition rules out assembly operations.

(5) A known, finite time is required to perform each operation and each operation must be completed before any operation which it must precede can begin (no "lap-phasing"). The operation time includes setup time.

(6) The time intervals for processing are independent of the order in which the operations are performed. (In particular, setup times are sequence-independent, and transportation time between machines is negligible.)

(7) In-process inventory is allowable.

(8) Machines never break down, and manpower of uniform ability is always available.

(9) Due dates are known and fixed.

(10) The job routing is given and no alternative routings are permitted.

(11) There is only one of each type of machine. (There are no machine groups.)

(12) All jobs are known and are ready to start processing before the period under consideration begins. (This is the "static" scheduling problem.)

Problems restricted only by the above set of constraints are sufficiently general to warrant attention. The difficulty is that such problems are also annoyingly invulnerable to most analytic approaches. While additional constraints, such as limitations on numbers of machines and/or jobs, permit easier analysis, the net effect is generally to reduce the original problem to a trivial special case, in which instance solution is no longer profitable.

Various Problem Approaches

The job shop scheduling problem has gone wanting a truly general solution, primarily because it is extraordinarily complex. In examples of realistic proportions, the sequencing issue alone has confounded the analysts. (A simple two-machine, three-job case has 36 possible sequences; a five-machine, six-job case has 720^5!)

Faced with this predicament, the courses open to the analyst are unclear. One alternative is optimization. This approach generally involves taking a problem of real dimensions and "whittling" it down by incrementally imposing constraints until it becomes a workable abstraction. If a digital computer is available, the simplification need be less extensive, but such devices are not always available. In their absence, the criticisms of the foregoing section are relevant: Problems for which optimal solutions are readily available are generally of little or no interest.

A second alternative is suboptimization. Instead of striving for a perfect solution to an abstraction, the analyst may attempt to develop a highly refined working approach to life-size problems that may not guarantee optimality but works reasonably well most of the time.

In the literature, the above alternatives have had their counterparts in five major approach areas, four of which will now be discussed.

Enumerative-combinatorial approach. In relatively simple situations (say two machines, three jobs), enumeration is a reasonable method of attack. It represents a straight-forward means of arriving at a guaranteed optimal answer to the sequencing problem. This advantage, however, is offset by computational inefficiency, which renders the method quite useless in most realistically complex situations.

The enumerative approach was originally used by Jackson,[5] Johnson,[6] Bellman,[7] Smith,[8] Mitten,[9] and McNaughton.[10] These studies generally confined themselves to cases involving three or fewer stages with a single machine at each stage, or a single stage with multiple identical machines. More elaborate cases were considered by Giffler and Thompson[11] and Heller,[12] but again the practicality of enumeration was in question. (Giffler and Thompson, in fact, suggested that sampling techniques be used in large-scale examples.) More recent work by Dudek and Teuton,[13] Gapp,[14] Root,[15] Smith and Dudek,[16] Gupta[17] and Florian et al.,[18] has generally embraced a modified combinatorial approach in which artificial restrictions are used to reduce the search.[19]

Mathematical programming. (a) Integer Linear Programming. Integer linear programming has been suggested by several authors, notably Bowman,[20] Wagner[21] and Manne.[22] Optimality is guaranteed, but again the difficulty is computational impracticality. A simple problem (say four machines, three products) requires 31 variables and 94 constraints by Wagner's formulation, and many more by the others. In an application study, Wagner himself has pointed to the need for deriving "methods which more fully take into account the special structure of machine sequencing problems."

In 1963, Little et al.,[23] developed a "branch-and-bound" algorithm for the traveling salesman problem.[24] This was applied to the three-machine case by Lomnicki,[25] and to the four-machine case by Ignall and Schrage.[26] Further applications were discussed by Gilmore and Gomory,[27] Brown and Lomnicki,[28] McMahon and Burton,[29] Greenberg,[30] Charlton and Death,[31] Maxwell,[32] and Florian, Trepant and McMahon.[33]

(b) Dynamic Programming. Dynamic programming, as an approach to the handling of scheduling problems, was suggested in 1956 by Bellman[34] and in 1962 by Held and Karp.[35] Bellman[36] used it to solve the traveling salesman problem, and Held et al.[37] applied it to assembly-line balancing. It was used by Wagner and Shetty,[38] by Bomberger,[39] by Presby and Wolfson,[40] by Moore,[41] by Lawler and Moore,[42] and by Glassey.[43] Of these, the Moore attempts to treat a situation with many jobs (>30) by partitioning these into two subsets, an operation which improves computational efficiency. The Lawler and Moore uses an equation similar to that for the "knapsack problem," and treats a variety of single-machine scheduling problems.

(c) Basic Linear Programming. In 1959, Dantzig et al.[44] used a joint linear programming-combinatorial approach to the traveling salesman problem. This methodology was later applied to a simplified m-machine, n-job case.[45] Finally, a bivalent ("zero-one") programming model was developed by von Lanzenauer.[46]

A weakness of basic linear programming is that its use may lead to fractional optimal solutions that are physically illogical, e.g. one half of a job processed in one fashion, the other half in another. Because integer programming overcomes this difficulty, it has largely supplanted pure linear programming in current usage.

Monte Carlo sampling and simulation. Basically, the Monte Carlo method involves sampling from a population of feasible solutions (schedules), the intention being to apply principles of statistical inference to the sample. Selection of the "best" schedule from a given sample does not guarantee that other "better" schedules do not exist outside the sample, but the likelihood of this possibility can be made arbitrarily small.

The sample of feasible solutions may be generated from various "runs" of a job shop simulator. This technique normally requires construction of a computerized model, which is used to duplicate (numerically) actual job shop conditions.

Heller and Logemann[47] and Giffler and Thompson[48] both applied Monte Carlo sampling in cases that were previously approached from a purely enumerative standpoint. Computer simulation studies were undertaken by Jackson,[49] Baker and Dzielinski,[50] Gordon,[51] Legrande,[52] and Bulkin et al.[53] The last of these is an interesting study of production control in a fabrication shop at Hughes Aircraft.

Graphical and miscellaneous approaches. (a) Graphical. A graphical, non-numerical approach to the scheduling problem was first suggested by Akers and Friedman.[54] Joint dynamic programming-graphical techniques were subsequently introduced by Szwarc.[55] Hardgrave and Nemhauser[56] devised a graphical version of Giffler and Thompson's early Gantt chart approach, enabling them to find minimum total processing time in the two-machine case. Finally, a precedence graph algorithm was concocted by Ashour and Parker.[57]

(b) Mixed. Several studies have used combinations of various programming techniques. Elmaghraby,[58] for example, uses linear, integer, and dynamic programming methods to handle the loading problem. Emmons[59] also uses a combination of techniques in his analysis of one-machine sequencing.

Methodological Weaknesses

It has been suggested that the more mathematically rigorous an approach, the more likely it is to encounter difficulty in practical application. To illustrate the validity of this statement, consider, for example, the linear programming interpretation of the scheduling problem proposed by Bowman.[60] Bowman's suggested approach judiciously adheres to all of the rigorously defined precepts of linear programming and, in doing so, quickly becomes self-defeating.

Bowman's problem involves three jobs, X, Y and Z, and four machines, A, B, C and D. The established machine sequence for job X is (ABCD); for job Y, (CADB); and for job Z, (DA). (Job Z requires no processing on machines B and C.)

The basic variables in the problem are of the form j(m,i) where j refers to the job in question, m refers to the operation, and i refers to the time period during which the operation takes place. (For example, X(A,1) stands for job X, machine A, and time period 1.)

Variables can have a value of either zero or one. If, for example, X(A,1) has value zero, the interpretation is that no work is being performed on job X by machine A during time period 1. Alternatively, a value of unity means that work <u>is</u> being performed.

The problem requires specification of several different sets of constraints. The first set accounts for the fact that all variables must have values of either zero or one at all times, that is, during a given time period, a process is either taking place or not. Hence:

$$1 \geq X(A,1), X(A,2), \ldots, X(A,T), X(B,1), X(B,2), \ldots,$$
$$Y(A,1), \ldots, Y(D,T), \ldots, Z(D,T) \geq 0.$$

The second set of constraints ascertains that all individual operations will be performed in their entireties. (e. g. product X requires 5 time units on machine A). Examples of these constraints are:

$$\Sigma_{i=1}^{i=T} X(A,i) = 5, \quad \Sigma_{i=1}^{i=T} X(B,i) = 2,$$

$$\Sigma_{i=1}^{i=T} Y(A,i) = 4, \quad \Sigma_{i=1}^{i=T} Z(D,i) = 6.$$

The third set of constraints ascertains that no two jobs will be processed by the same machine at the same time. Constraints are of the form:

$$X(A,1) + Y(A,1) + Z(A,1) \leq 1$$
$$X(A,2) + Y(A,2) + Z(A,2) \leq 1$$
$$\vdots$$
$$X(D,T) + Y(D,T) + Z(D,T) \leq 1$$

The fourth set dictates adherence to sequencing requirements, specifically, no operation can begin until the previous operation (as prescribed in the sequence) has been completed. As an example, job X must be processed for 5 time units on machine A before it can begin processing on machine B. Similarly, the job must be processed for 2 time units on machine B before it can be started on C. Thus,

$$5X(B,j) \leq \Sigma_{i=1}^{i=j-1} X(A,i)$$

$$2X(C,j) \leq \Sigma_{i=1}^{i=j-1} X(B,i)$$

$$6Z(A,j) \leq \Sigma_{i=1}^{i=j-1} Z(D,i)$$

for all j from 1 to T.

To prevent interruption prior to completion of any operation, a fifth set of constraints is added. The constraints are of the form:

$$5X(A,i) - 5X(A,i+1) + \Sigma_{j=i+2}^{j=T} X(A,j) \leq 5$$

$$2X(B,i) - 2X(B,i+1) + \Sigma_{j=i+2}^{j=T} X(B,j) \leq 2$$

.

.

.

$$6Z(D,i) - 6Z(D,i+1) + \Sigma_{j=i+2}^{j=T} Z(D,j) \leq 6$$

for all i from 1 to T.

Upon analysis, the Bowman formulation is found to require between 300 and 600 real variables, as well as a substantially greater number of constraints. Clearly, this represents a formidable amount of data, especially when one considers the limited scale of the problem (3 jobs and 4 machines). On this basis, problems of any reasonable size would be expected to be entirely unworkable.

Considerable advances have been made since Bowman originally delineated the linear programming approach to job shop scheduling in 1959. Specialized types of programming, such as "Zero-One," combined with "branch-and'bound" techniques, have greatly improved computational efficiency, yet the disadvantages of rigorous, formal analysis remain. The problem-solving process has been made simpler, yet not so simple as to permit truly efficient generation of precisely optimal solutions to very large problems, even assuming access to the most advanced kinds of computational equipment.

Linear programming is but one example of a formal, analytic approach to the scheduling problem. In practice, enumeration and combinatorial techniques are even less successful because they make even greater demands on computational facilities.

Simulation techniques have been used with some success, although accurate modeling of the job shop is likely to be a formidable task. Each operation tends to require a specialized simulation, which precludes the use of general purpose software "packages." Thus, heavy development costs in combination with high costs of the hardware itself generally prohibit simulation as a feasible approach in all but the largest job shops.

The answer to all these difficulties would appear to lie in the use of a less precise methodology which is not oriented toward extreme accuracy of results. One such methodology involves the use of dependable "heuristics." Before proceeding to delineate the advantages of this approach, however, it is first necessary to examine the various interpretations that the term has received in the literature.

Definition of Heuristics

Webster's Third New International Dictionary defines the adjective "heuristic" as "serving to guide, discover, or reveal; specif.: valuable for stimulating or conducting empirical research but unproved or incapable of proof- often used of arguments, methods or constructs that assume or postulate what remains to be proven or that lead a person to find out for himself."[61] Similarly, the Random House Dictionary of the English Language defines heuristic as (1) "serving to indicate or point out; stimulating interest as a means of furthering investigation" or (2) "encouraging the student to discover for himself."[62]

In common business usage, the term "heuristics" has received various specialized definitions, each of which tends to emphasize a particular aspect of the basic meaning. For example, many authors[63] equate heuristics with "rules of thumb," defined by Webster's as "method(s) of procedure or analysis based upon experience and common sense and intended to give generally or approximately correct or effective results."[64] While connotations of "guidance" are clearly present in both definitions, the latter is much more explicit in stressing "experience," "common sense," and outcomes that are only "approximately correct or effective."

Another group of authors, following the interpretation of Simon and Newell,[65] stress the connotation of "self-discovery" or learning.[66] (Simon and Newell first applied the word "heuristic" to instances in which computers could be programmed to duplicate human intelligence, in the sense of problem solving and the development of strategies.)

Still a third group of researchers, suggesting alliance with Simon and Newell, have sought to define heuristics purely in terms of search reduction.[67] For example, any approach to the sequencing problem requiring less than full enumeration of all possible sequences is said to be heuristic. Thus, all "branch-and-bound" and similar algorithmic approaches would properly be included within this definition. (Note that while search limitation is part of the heuristic problem-solving process as defined by Simon and Newell, it does not appear as a part of the basic definition of "heuristic.")

As Weist aptly points out, it appears that common usage permits a heuristic to be defined as "any systematic way of solving problems- e.g., systematic cut-and-try based on reasonable rules of thumb at one extreme, and algorithms with their supporting theories and known properties at another extreme."[68] Indeed, this variability of meaning is clearly documented in the next section.

Job Shop Heuristics: Summary of the Literature

One of the earliest appearances of the word "heuristic" in scheduling literature was in connection with a line-balancing problem described by Tonge.[69] Although Tonge's study did not bear directly on the job shop problem, it was significant in that it served to test the feasibility of a heuristic approach to complex decision-making. On the basis of the outcome, Tonge concluded that heuristics was a valuable scheduling tool.

The basic line-balancing problem is as follows: Given an assembly process made up of elemental tasks, each with a time required per unit of product and an ordering with other tasks, what is the least number of work stations needed to obtain a desired production rate? To answer this question, Tonge proposed a heuristic procedure consisting of three phases:

(1) Repeated simplification of the initial problem by grouping adjacent elemental tasks into compound tasks.

(2) Solution of the simpler problems thus created by assigning tasks to work stations at the least complex level possible, breaking up the compound tasks into their elements only when necessary for a solution.

(3) Smoothing the resulting balance by transferring tasks among work stations until the distribution of assigned tasks is as even as possible.

An early major study investigating artificial learning processes in a job shop setting was that of Fischer and Thompson.[70] The fundamental conjecture was that probabilistic selection of loading rules is initially superior to any other mode of selection, but that learning can be used to eventually improve the selection process.

To test this conjecture, an "unbiased random process" was used to select a loading rule from a given set of rules. (The experimental set consisted of only two rules, the SIO ("shortest imminent operation") and LRT ("longest remaining time"), although any number might have been included.) Whenever it was necessary to decide which of several jobs should be scheduled next on a given machine, a new "drawing" was made.

Fischer and Thompson succeeded in demonstrating the validity of the first part of the conjecture: The unbiased random process did reasonably well as a scheduling heuristic and invariably produced better results than the "worst" rule in the set. The second part of the conjecture, however, was not so clearly borne out.

It was felt that experience with probabilistic selection of rules might lead to the formulation of a more systematic method of choice that could improve overall results. (For example, it seemed reasonable that the SIO rule should be employed in early scheduling decisions and the LRT rule in later ones.) A learning program was thus designed and tested.

The remainder of the Fischer-Thompson article contains a description of specific learning processes incorporated into the program, as well as elaborate statistical testing of results. The discouraging conclusion, drawn from the findings, is that learning is possible, but not very desirable in light of the (statistically) very minor advantages that accrue. The additional effort required to incorporate a learning device into the random selection process simply did not appear to be justified.

Somewhat more optimistic results were obtained by Crabill.[71] Examining what he calls a "job-at-a-time" adjusting procedure, Crabill's method calls for scheduling each operation of a particular job in order to optimize its effect on the maximum flow-time of all jobs. Once an operation is tentatively inserted into "best" position, the effect on previously scheduled operations is determined by computing new operation starting times and other relevant data.

Specifically, the "best" position for a particular operation of job J is the one that minimizes a lower-bound estimate of job flow-time over all jobs. Assume T represents the potential starting time of job J on machine I. Also assume j=1,2,...,n represents the sequenced set of jobs scheduled on machine I, but not completed at time T. (There are n+1 positions available for the "insertion" of job J, before the first job, or immediately following each of the n scheduled jobs.) Finally, let D_j be the current finish time of job j on machine I; C_j be the current completion time of job j; C_j be the smallest completion time of job j; E_{jq} be the finish time of job j (J,1,2,...,n) on machine I <u>if</u> job Q is inserted into schedule position q=(0,1,...,n); and t_J be the processing time of job J on machine I.

For every schedule position q, Crabill computes the quantity: $\max(C_J+E_{Jq}-T-t_J, \max (C_j+E_{jq}-D_j))$ where j=1,..., n. The schedule position that minimizes this quantity is selected as the position for job J.

Note that if any job j is completed beyond its current finish time on machine I (D_j), its overall completion time may be increased by approximately $E_{jq}-D_j$. (This bound is imprecise because waiting time at subsequent operations tends to reduce it, while precedence constraints tend to amplify it.) If started in position q, the waiting time of job J is $C_J+E_{Jq}-T-t_J$. The position selected is the one that minimizes the increase in lower-bound completion time for any job.

In Crabill's method, "learning" proceeds from an initial schedule in which jobs are inserted in decreasing order of total processing time. Twenty problems were considered, each requiring nine applications of the adjusting procedure. The results compared quite favorably with best outcomes obtained through the use of dispatching procedures.

In 1964, Karg and Thompson devised a heuristic approach to solving traveling salesman problems. This was of special significance because of the close relationship between job shop sequencing and the traveling salesman.[72]

The classic traveling salesman problem may be described as follows: A salesman must visit each of a given number of cities exactly once before returning home. Locations of all cities are specified, and the object of the problem is to determine the closed-loop route (or routes) that minimize the total distance traveled. (In the analogous machine shop sequencing problem, the cities are the machines, and the inter-city distances are the times spent at each machine. Minimization of total processing time is akin to minimization of total distance traveled.)

When the number of cities is small, the problem can easily be solved, either by numerical enumeration, by integer programming, or by graphical analysis. When the number of cities is very large, however, these methods become impractical.

The heuristic approach is used to obtain an optimal or near-optimal solution in instances that cannot easily be solved by more mathematically rigorous means. Given a listing of cities a heuristic rule is used to generate a restricted set of closed loop routes that may or may not contain the optimal route. The shortest route is then picked from this set. The properties of this shortest route are then used to define sub-problems, which are subsequently attacked in the same way as the original problem.

Schwartz considers the n-operation, two-machine sequencing problem in terms of precedence diagrams and assignment matrices.[73] The model permits variable operation times, as well as utilization of either or both machines. A similar approach was used earlier to treat a less flexible m-machine case.[74]

The theory of precedence diagrams is intimately related to that of critical path schedules, PERT diagrams, and Gantt charts. The precedence diagram is a display of technological ordering restrictions in which numbered nodes are used to designate the operations, and arrows between the nodes are used to indicate precedence. Operation times and machine codes are also included in the diagram.

For computational purposes, Schwartz converts the precedence diagram to an assignment matrix. When a computer is used, nodes are designated by index numbers, and precedence is indicated by the positioning of Boolean elements throughout the matrix. (For example, if node i must precede node j, a "1" appears in the jth row, ith column.) When problems are to be solved manually, the Boolean element "1" is replaced with its node designation, and the Boolean element "0" is omitted.

Schwartz's heuristic method can best be illustrated with reference to an example. Consider the assignment matrix (Figure 1) which was derived from the precedence diagram (Figure 2). Note carefully the last two columns of the assignment matrix. These contain processing times, required for each operation, on machines A and/or B.

The following definitions pertain:

(1) Precedence number. The precedence number specifies the number of operations that follow any given operation. Precedence numbers for each operation are obtained by counting occupied grids in each column of the assignment matrix. For example, column 1 has six occupied grids, so 6 other operations must follow operation 1. Furthermore, these operations are 5, 9, 13, 17, 20 and 23, as may be determined from the left-hand scale of the assignment matrix.

(2) Precedence time. The precedence of an operation is the total time required to process all <u>unassigned</u> operations that follow it. Precedence time for a given operation is thus obtained by adding the A-B column figures appearing in only those rows which correspond to subsequent operations. For example, operations 19, 22 and 24 follow operation 16

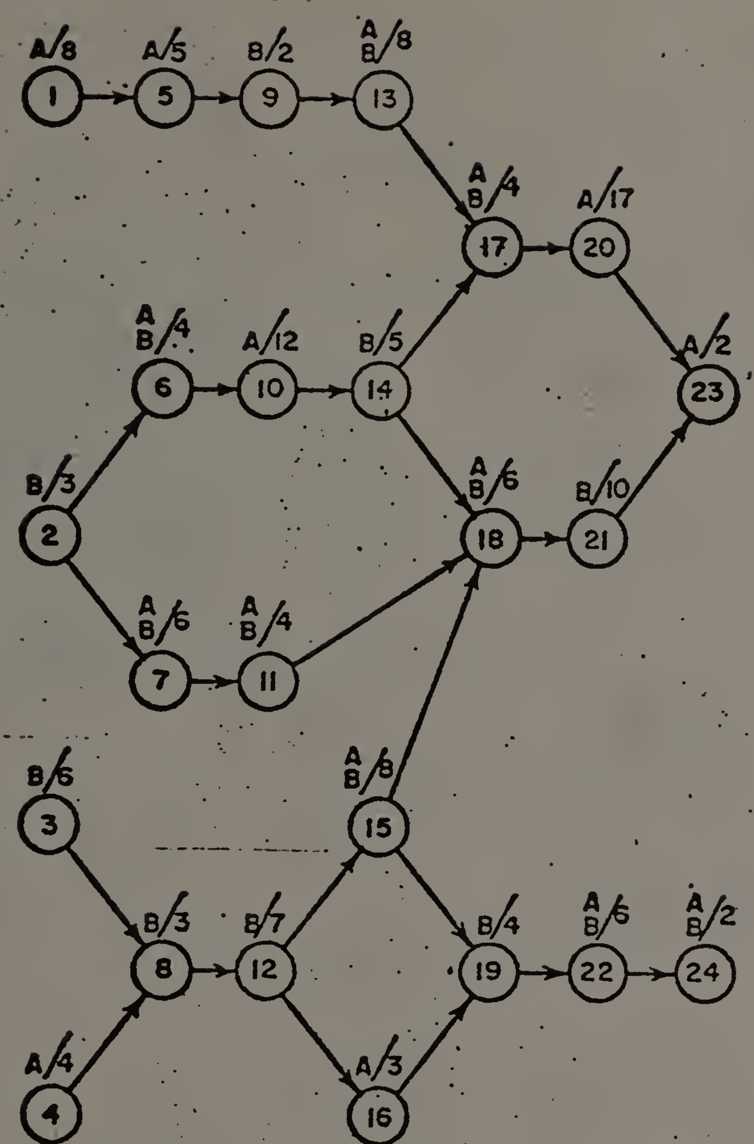


Fig. 1

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	A	B
1																									8	
2																										3
3																										6
4																									4	
5	1																								5	
6		2																							2	2
7		2																							3	3
8			3	4																						3
9	1				5																					2
10		2				6																			12	
11		2					7																		2	2
12			3	4				8																		7
13	1				5				9																4	4
14		2				6				10																5
15			3	4				8				12													4	4
16			3	4				8				12													3	
17	1	2			5	6			9	10			13	14											2	2
18		2	3	4		6	7	8		10	11	12		14	15										3	3
19			3	4				8				12			15	16										4
20	1	2			5	6			9	10			13	14			17								17	
21		2	3	4		6	7	8		10	11	12		14	15			18								10
22			3	4				8				12			15	16			19						3	3
23	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15		17	18		20	21				2	
24			3	4				8				12			15	16			19			22			1	1
																									75	64

Fig. 2

and require a total of 4+3+3+1+1=12 time units. The components of the addition are read from columns A and B, rows 19, 22 and 24.

The procedure involves three basic steps: uncovering operations, assignment, and ensuring feasibility.

(3) Uncovered operations. Uncovered operations have no predecessors and may therefore be assigned freely. An uncovered operation creates a blank row in the assignment matrix. In Figure 2, operations 1, 2, 3, and 4 have no predecessor operations, and their rows are therefore blank. Operation 5, on the other hand, is preceded by operation 1, operation 6 by operation 2, and so on.

The entire sequencing procedure may be summarized as follows: The assignment matrix is prepared and then inspected for uncovered operations. Operation 1 will always be uncovered. If it appears in isolation, it will be assigned immediately to a machine. On the other hand, several uncovered operations may appear simultaneously, in which case it will be necessary to determine priorities for their assignment.

Let T_A and T_B be total elapsed running times of machines A and B at the time the priority determination must be made. If these are equal, then the operation with the greatest precedence time is selected for assignment. This assures the uncovering of those operations with the greatest total remaining time. However, if T_A and T_B are not equal, the operation whose machine has the minimum accumulated time should be chosen. Such a decision leads to minimization of idle time in the following period.

Once an operation has been disposed of, the associated columnar elements are reduced to zero. This creates at least one new blank row and, hence, one new uncovered operation. (The reader may verify that elimination of column 1 invariably results in the uncovering of operation 2.) Hence, new uncovered operations are constantly being generated.

Finally, uncovered operations are placed in storage and sorted into sets of fixed and flexible assignments. Additionally, uncovered oeprations with fixed assignments are assigned to designated machines. (The handling of flexible assignments is somewhat involved and will not be dealt with here. However, such omission in no way interferes with Schwartz's basic heuristic argument.)

In terms of theory, Schwartz's assignment rules seek to minimize idle time and balance the parallel sequences. This requires examination of current elapsed time on each machine, as well as remaining running time of covered operations. Ideally, the total set of operations would be equally divided between the two machines. It is therefore possible to compute a lower bound for total elapsed time of the parallel sequences (although this lower bound may not necessarily be a feasible solution.)

Actual application of Schwartz's method reveals that the algorithm tends to converge more closely to an optimum (lower bound) as the number of operations increases. This occurs because the number of permutations on operations increases and several optimum solutions may exist. The scope of the problem affords many opportunities to make adjustments. Thus a bad decision at one point can be offset by a good decision at a subsequent point.

In 1965, Gavett developed three heuristic rules as a means of approaching the single machine sequencing problem. The objective of the sequencing decision was to minimize machine setup time over a set of heterogeneous jobs.[75]

Under completely deterministic circumstances, it is theoretically a simple matter to minimize total machine setup time. The procedure involves constructing a matrix of t_{ij}s, where each t_{ij} is the time required to change the facility (machine) from processing job i to processing job j. (t_{ij} is not necessarily equal to t_{ji}.) If the number of jobs is relatively small, the job sequence requiring minimum total setup time can be found by direct enumeration of all possible sequences. If the number of jobs is relatively large, the optimal sequence can be obtained by applying a variation of the classic "branch-and-bound" approach to the "traveling salesman" problem.[76]

Gavett is interested in those situations which do not precisely conform to the above description, and, consequently, into which it is desirable to introduce heuristics. In particular, he aptly points out that the paired-job setup times, t_{ij}, are frequently difficult to estimate accurately. This fact often negates the validity of enumerative or "branch-and-bound" solutions.

In large-scale problems, the application of "branch-and-bound" or other algorithmic techniques frequently places heavy demands on computational facilities. If such facilities are small or non-existent, the use of such techniques is often highly impractical. This is, of course, another point in favor of simple heuristic rules.

Gavett suggests three heuristic rules:

(1) The "next best" rule. "Always select the unassigned job which has the least setup time relative to the job which had just been completed." Upon selection of a particular job for processing, the rule is applied to the remaining set of jobs. This process continues until all jobs are exhausted.

The "next best" rule is intuitively compelling because it suggests that the machine operator, having some knowledge of job characteristics, will always schedule similar jobs as a group, in order to minimize increases in setup time from one job to the next. In many instances, this appears to be a reasonably valid point of view.

(2) The "next best" rule with variable origin. This rule initially calls for the application of the "next best" rule, in order to determine the first job in the sequence.

It then calls for trial insertion of all possible jobs in second sequence position, the "next best" rule subsequently being applied in each case.

To illustrate this semi-enumerative rule, consider an initial sequence 1-6-2-3-5-4. The first two jobs of subsequent trial sequences will be 1-2, 1-3, 1-4, and 1-5. The remainder of each sequence will be generated by individual application of the "next best" rule, and the sequence chosen will be the one for which total setup time is a minimum. (Note that this method requires the construction of a setup time matrix.)

(3) The "next best" rule after column deductions. For any job J, this rule calls for the reduction of each t_{iJ} by an amount equal to $\min(t_{1J}, t_{2J}, \ldots, t_{nJ})$. The "next best" rule is then applied to obtain a feasible sequence.

To test these rules, a sample of setup-time matrices was generated by a computer. The heuristic rules were applied to each matrix, and results were compared with (1) optimum values obtained through application of the "branch-and-bound" algorithm, and (2) values obtained through random sampling. The conclusion was that the "next best" rule and its variants represented a significant improvement over random sequencing.

In the testing, setup times were not held precisely rigid, but were assumed flexible over a small range of values. The evidence was that small variances did not seriously interfere with the general usefulness of the heuristic rules.

Heuristic Approaches to Date: An Appraisal

The studies cited in the foregoing section are believed to represent a good cross-section of the heuristic literature. They easily verify Weist's contention that the term "heuristic" has been subjected to an enormous range of interpretations.

It is not our intention to pass judgment on the liberties that authors have taken in defining the term, but merely to point out that some interpretations appear to be less useful than others. If the primary purpose of the heuristic approach is held to be simplification, then there is some question as to whether the more highly refined algorithmic approaches achieve their goal. Clearly, many of these approaches do not permit manual generation of solutions and, what is worse, many of them require computational facilities that could hardly be described as modest.

In addition to this, the logic underlying many of the approaches is extremely difficult to discern, and, in fact, many authors seem totally unwilling to engage in any type of interpretive discussion. Under these conditions, advocates of a given scheme would be expected to proselytize a limited number of theoreticians, but with little or no hope of ever developing much of a following from the ranks of the practitioners. This is indeed unfortunate, since practical application should evidently be a matter of first priority.

In order to insure maximum acceptability within the field, heuristic methods should evidently be oriented towards ease of application and simplicity of interpretation. The first goal can be achieved by defining "heuristics" in line with the "rule of thumb" interpretation; the second can be accomplished by stressing the logic underlying every aspect of the proposed approach.

A close examination of the literature reveals that the bulk of work dealing with job shop heuristics of the "rule of thumb" variety seeks to subject certain priority rules to simulative testing. The general approach has been to start with some of the more basic rules ("job slack," "first-come, first-served," "shortest imminent operation," etc.) and then to develop and test more sophisticated rules. At any rate, the emphasis has been on the testing of rules in order to determine their effectiveness, rather than on practical application and intuitive understanding.[77]

Curiously, there have apparently been no popular expositions which seek to explain how one actually goes about successfully employing priority rules in an actual, job-shop setting. The simulation studies have been reasonable effective in determining the relative usefulness of certain simple and compound rules, but the intelligent application of such rules is a matter left to question. Baker and Dzielinski, for example, found that the "shortest imminent processing time" rule worked well if it was desired to minimize the average of all jobs' manufacturing times.[78] But how, precisely, does one take that knowledge and apply it in actual scheduling practice? This is the topic to which we wish to address ourselves in the dissertation.

Basic Approach of the Dissertation

In order to arrive at a satisfactory solution to the job shop scheduling problem, we propose a highly flexible intuitive approach that is not limited by the dynamic nature of the scheduling environment. The method, in fact, focuses on adaptability as the most important single characteristic of the job-shop scheduler.

Of extreme benefit to the small job shop is the procedure's modest requirement in terms of both human and machine capability. A basic understanding of Gantt charts, as well as some "feel" for the logic behind priority rules, would be considered important, but, beyond that, the approach presupposes no specialized skills or aptitudes. In addition, any required calculations can normally be handled with paper and pencil.

The discussion will center around several prototypic examples. In each example, several jobs must pass through a set of machines in some prescribed sequence, although this sequence may vary from job to job.[79] The processing time required by each job at each work station is fixed and known.

Setup and transferral times are assumed to be included in processing times, and assembly times, if any exist, are assumed negligible.

In the initial formulation of the problem, additional restrictions preclude cancellation or addition of orders during the period over which the original job set is to be processed (i.e. the situation is "static."). Once the analytic framework has been established, however, these constraints will be dropped, that is, the scheduling environment will become "dynamic."

The approach requires that the sequencing of each job be tentatively determined according to one of several simple priority rules (to be discussed shortly). The choice of priority rules is dependent, to a degree, on the criteria to be "optimized," but even the most careful rule selection does not guarantee a satisfactory schedule at this point. If improvement is desired, adjustment of the original working schedule may be accomplished by applying one or more of several supplementary heuristic rules.

Objectives. Initially, the objective is to devise a schedule that results in "minimum" schedule length (makespan), or, alternatively, in "minimum" idle time. In a later section, critical dates (due dates) are introduced. The objective is then to construct a schedule that permits all jobs to be completed "on time," or, alternatively, with minimum lateness.

The use of quotation marks on words such as "minimum," "maximum" and "optimum" is meant to suggest that the terms are only approximately valid in a context of this sort. A true heuristic methodology by nature is incapable of guaranteeing an optimal solution to any problem, although exactly optimal solutions may occur by accident. The best that can be expected is a solution that is very near optimal, although the likelihood of this diminishes somewhat as problem size grows beyond the limits of human assimilative ability. (The value of methodologies that "suboptimize" rather than optimize will be discussed subsequently.)

Priority rules. In the dissertation, several of the more commonly encountered priority rules will be considered. The application of these rules will be illustrated, using each of the sample problems as an example.

The following will be discussed:

(1) First-come, first served. This is the classic rule which affords first priority to first arrivals.

(2) Shortest imminent processing time. According to this rule, orders are arranged such that the job requiring the least processing time on the very next operation is scheduled first. This rule is of particular interest since its use in simulations has produced some very good results.[80]

(3) Longest imminent processing time. According to this rule, orders are arranged such that the job requiring the greatest processing time on the very next operation is

scheduled first.

(4) Job slack. Maximum priority is given to the job which has the smallest difference between time remaining to its due date and total remaining processing time.

(5) Job slack- remaining operations. This rule establishes priority on the basis of the ratio S/N, where S = slack and N = the number of operations remaining beyond the decision point. The smaller the ratio of S to N, the higher the priority of the job in question.

(6) Job slack- remaining time. This rule asserts that the priority of jobs should be judged on the basis of slack in relation to total remaining pre-deadline time. Specifically, the measure is S/T; the smaller this ratio, the higher the priority of the associated job.

Secondary rules. The key to successful application of priority rules is believed to lie in the scheduler's ability to supplement them with dependable heuristics. Frequently, for example, a schedule that fails to meet due-date requirements can be successfully adjusted by applying secondary heuristics, such as "If scheduling according to the primary rule causes a job to become late, consider the reassignment of priorities" or "If obvious gaps exist, try to fill them." These and similar rules will be discussed extensively in the course of the dissertation.

A Philosophical Note

Even the most carefully designed heuristic approach is subject to criticism by those who maintain that the only useful solution to any problem is the optimal one. This commonly-encountered argument has not, however, gone without rebuttal. For example, in their book Organizations, March and Simon state that

> "...finding the optimal alternative is a radically different problem from finding a satisfactory alternative. An alternative is optimal if: (1) there exists a set of criteria that permits all alternatives to be compared, and (2) the alternative in question is preferred, by these criteria, to all other alternatives. An alternative is satisfactory if: (1) there exists a set of criteria that describes minimally satisfactory alternatives, and (2) the alternative in question meets or exceeds all these criteria.
>
> Most human decision-making, whether individual or organizational, is concerned with the discovery and selection of satisfactory alternatives; only in exceptional cases is it concerned with the discovery and selection of optimal alternatives. To optimize requires processes several orders of magnitude more complex than those required to 'satisfice'. An example is the difference between searching a haystack to find the sharpest needle in it and searching the haystack to find a needle sharp enough to sew with."[81]

The proposed heuristic approach to shop scheduling is very accurately definable in terms of "satisficing." A limited number of schedules is generated, and each is evaluated on the basis of a specific "criterion." The schedule which yields the best results, in terms of that criterion, is the one selected.

Linear programmers might argue that March and Simon's "criteria" are actually constraints, since they are fixed and not capable of being optimized. In a linear programming context, they would, of course, be correct. Note, however, that the non-specialized definition of "criterion" is merely "a standard of judging," and this is precisely what is implied in the foregoing context.

In any instance where optimization is either impossible or impractical, "satisficing" would appear to be a highly attractive alternative, even in spite of the obvious trade-offs. In the following pages, we outline a heuristic approach to shop scheduling that is fully consistent with this outlook.

Footnotes

1. Roger L. Sisson, "Methods of Sequencing in Job shops-A Review," Operations Research, 7 (January, 1959), 10-29.

2. Ralph Bravoco, "The Scheduling (Sequencing) Problem-Review and Extensions" (unpublished paper, University of Massachusetts, 1971).

3. Webster's Third New International Dictionary (Springfield, Mass.: G.C. Merriam Company, 1967), p. 2028.

4. William S. Gere, "Heuristics in Job Shop Scheduling," Management Science, 13 (November, 1966), 167-190.

5. J.R. Jackson, "An Extension of Johnson's Results of Job Lot Scheduling," Naval Research Logistics Quarterly, 3 (September, 1956), 201-203.

6. S.M. Johnson, "Optimal Two and Three-Stage Production Schedules with Setup Times Included," Naval Research Logistics Quarterly, 1 (March, 1954), 61-68.

7. Richard Bellman, "Mathematical Aspects of Scheduling Theory," Journal of the Society of Industrial and Applied Mathematics, 4 (1956), 168-205.

8. W.E. Smith, "Various Optimizers for Single Stage Production," Naval Research Logistics Quarterly, 3 (1956), 59-66.

9. L.G. Mitten, "Sequencing n Jobs on Two Machines with Arbitrary Time Lags," Management Science, 5 (1959), 293-298.

10. R. McNaughton, "Scheduling with Deadlines and Loss Functions," Management Science, 6 (1959), 1-12.

11. B. Giffler and G.L. Thompson, "Algorithms for Solving Production Scheduling Problems," Operations Research, 8 (July, 1960), 487-503.

12. J. Heller, "Some Problems in Linear Graph Theory that Arise in the Analysis of the Sequence of Jobs through Machines" (AEC Research and Development Report NYO-987, 1960).

13. R.A. Dudek and O.F. Teuton, "Development of M-Stage Decision Rule for Scheduling n Jobs through M Machines," Operations Research, 12 (May, 1964), 471-497.

14. W. Gapp, P.S. Minkekar, L.G. Mitten, "Sequencing Operations to Minimize In-Process Inventory Costs," Management Science, 11 (January, 1965), 476-484.

15. J.G. Root, "Scheduling with Deadlines and Loss Functions on K Parallel Machines," Management Science, 11 (January, 1965), 460-475.

16. R.D. Smith and R.A. Dudek, "A General Algorithm for Solution of the n Job, M Machine Sequencing Problem of the Flow Shop," Operations Research, 15 (January, 1967), 71-82.

17. J.N.D. Gupta, "Improved Combinatorial Algorithm for the Flow Shop Scheduling Problem," Operations Research, 19 (November, 1971), 1753-8.

18. M. Florian et al., "Implicit Enumeration Algorithm for the Machine Sequencing Problem," Management Science, 17 (August, 1971), B782-B792.

19. Some would argue that this approach is properly termed "heuristic." See ensuing discussion.

20. E.H. Bowman, "The Scheduling-Sequencing Problem," Operations Research, 7 (Sept., 1959), 621-4.

21. H.M. Wagner, "An Integer Linear Programming Model for Machine Scheduling," Naval Research Logistics Quarterly, 6 (June, 1959), 131-140.

22. A.S. Manne, "On the Job Shop Scheduling Problem," Operations Research, 8 (March, 1960), 219-223.

23. J.D.C. Little et al., "An Algorithm for the Traveling Salesman Problem," Operations Research, 11 (November, 1963), 972-89.

24. Some would argue that this approach is properly termed "heuristic." See ensuing discussion.

25. Z.A. Lomnicki, "A Branch-and-Bound Algorithm for the Exact Solution of the Three-Machine Scheduling Problem," Operational Research Quarterly, 16 (March, 1965), 89-100.

26. E. Ignall and L. Schrage, "Application of the Branch-and-Bound Technique to Some Flow Shop Scheduling Problems," Operations Research, 13 (May, 1965), 400-412.

27. P. Gilmore and R.E. Gomory, "Sequencing of One State-Variable Machine," Operations Research, 12 (September, 1964), 655-679.

28. A.P.G. Brown and Z.A. Lomnicki, "Some Applications of the Branch-and-Bound Algorithm to the Machine Scheduling Problem," Operational Research Quarterly, 17 (June, 1966), 173-186.

29. G.B. McMahon and P.G. Burton, "Flow Shop Scheduling with the Branch-and-Bound Method," Operations Research, 15 (May, 1967), 473-481.

30. H.H. Greenberg, "Branch-Bound Solution to the General Scheduling Problem," Operations Research, 16 (March, 1968), 351-61.

31. J.M. Charlton and C.C. Death, "Generalized Machine-Scheduling Algorithm," Operational Research Quarterly, 21 (March, 1970), 127-34.

32. W.L. Maxwell, "On Sequencing n Jobs on One Machine to Minimize the Number of Late Jobs," Management Science, 16 (January, 1970), 295-297.

33. M. Florian et al., "An Implicit Enumeration Algorithm for the Machine Sequencing Problem," Management Science, 17 (August, 1971), B782-B792.

34. Bellman, "Mathematical Aspects of Scheduling Theory," Journal of the Society of Industrial and Applied Mathematics, 4 (1956), 168-205.

35. M. Held and R.M. Karp, "A Dynamic Approach to the Sequencing Problem," Journal of the Society of Industrial and Applied Mathematics, 10 (1962), 48-53.

36. Richard Bellman, "Dynamic Programming Treatment of the Traveling Salesman Problem," Journal of the Association for Computing Machinery, 9 (January, 1962), 61-63.

37. H. Held et al., "Assembly-Line Balancing: Dynamic Programming with Precedence Constraints," Operations Research, 11 (May, 1963), 442-59.

38. D.W. Wagner and C.M. Shetty, "Solution to a Production Scheduling Problem with Fixed Costs," Operations Research, 13 (March, 1962), 87-94.

39. E.E. Bomberger, "A Dynamic Programming Approach to a Lot Size Scheduling Problem," Management Science, 12 (July, 1966), 778-784.

40. J.T. Presby and M.L. Wolfson, "Algorithm for Solving Job Sequencing Problems," Management Science, 13 (April, 1967), B454-B464.

41. J.M. Moore, "An n Job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs," Management Science, 15 (September, 1968), 102-109.

42. E.L. Lawler and J.M. Moore, "A Functional Equation and its Application to Resource Allocation and Sequencing Problems," Management Science, 16 (September, 1969), 77-84.

43. C.R. Glassey, "Dynamic Linear Programs for Production Scheduling," Operations Research, 19 (January, 1971), 45-56.

44.G. Dantzig et al., "On a Linear Programming Combinatorial Approach to the Traveling Salesman Problem," Operations Research, 7 (January, 1959), 58-66.

45. G. Dantzig, "A Machine Shop Scheduling Model," Management Science, 6 (January, 1960), 191-196.

46. C.H. Von Lanzenauer, "Production Scheduling Model by Bivalent Linear Programming," Management Science, 17 (September, 1970), 105-11.

47. J. Heller and G. Logemann, "An Algorithm for the Construction and Evaluation of Feasible Schedules," Management Science, 8 (January, 1962), 168-183.

48. B. Giffler et al., "Numerical Experience with the Linear and Monte Carlo Algorithms for Solving Production Scheduling Problems," in Industrial Scheduling, ed. by J.F. Muth and G.L. Thompson (Englewood Cliffs, New Jersey: Prentice-Hall, 1963).

49. J.R. Jackson, "Simulation Research on Job Shop Production," Naval Research Logistics Quarterly, 4 (December, 1957).

50. C.T. Baker and B.P. Dzielinski, "Simulation of a Simplified Job Shop," Management Science, 6 (1960), 311-323.

51. G. Gordon, "A General Purpose Systems Simulator," IBM Systems Journal, September, 1962.

52. Earl LeGrande, "The Development of a Factory Simulation System Using Actual Operating Data," Management Technology, 3 (May, 1963), 1-19.

53. M.H. Bulkin et al., "Load Forecasting, Priority Sequencing, and Simulation in a Job Shop Control System," Management Science, 13 (October, 1966), B29-B51.

54. S.B. Akers and J. Friedman, "A Non-Numerical Approach to Production Scheduling Problems," Operations Research, 3 (1955), 429-442.

55. W. Szwarc, "Solution of the Akers-Friedman Scheduling Problem," Operations Research, 8 (November, 1960), 782-788.

56. W.W. Hardgrave and G.L. Nemhauser, "A Geometric Model and a Graphical Algorithm for a Sequencing Problem," Operations Research, 11 (November, 1963), 889-900.

57. S. Ashour and R.G. Parker, "Precedence Graph Algorithm for the Shop Scheduling Problem," Operational Research Quarterly, 22 (June, 1971), 165-175.

58. S.E. Elmaghraby,"A Loading Problem in Process-Type Production," Operations Research, 16 (September, 1968), 902-914.

59. H. Emmons, "One-Machine Sequencing to Minimize Certain Functions of Job Tardiness," Operations Research, 17 (July, 1969), 701-715.

60. Bowman, "The Scheduling-Sequencing Problem," Operations Research, 7 (September, 1959), 621-4.

61. Webster's Third New International Dictionary, p. 1064.

62. Random House Dictionary of the English Language, college edition (New York: Random House, 1969), p. 622.

63. See, for example, Gere, "Heuristics in Job Shop Scheduling," Management Science, 13 (November, 1966), 167-190.

64. Webster's Third New International Dictionary, p. 1986.

65. H.A. Simon and A. Newell, "Heuristic Problem-Solving: The Next Advance in Operations Research," Operations Research, 6 (1958), 1-10.

66. See, for example, H. Fischer and G.L. Thompson, "Probabilistic Learning Combinations of Local Job Shop Scheduling Rules," in Industrial Scheduling, ed. by J.F. Muth and G.L. Thompson (Englewood Cliffs, New Jersey: Prentice-Hall, 1963).

67. M.S. Bakshi and S.R. Arora, "The Sequencing Problem," Management Science, 16 (December, 1969), B247-B263.

68. J.D. Weist, "A Heuristic Model for Scheduling Large Projects with Limited Resources," Management Science, 13 (February, 1967), B359-B377.

69. F.M. Tonge, "Summary of a Heuristic Line-Balancing Procedure," Management Science, 7 (October, 1960), 21-42.

70. Fischer and Thompson, "Probabilistic Learning Combinations of Local Job Shop Scheduling Rules," in Industrial Scheduling.

71. T.B. Crabill, "A Lower-Bound Approach to the Scheduling Problem," (research report, Department of Industrial Engineering, Cornell University, 1964).

72. R.L. Karg and G.L. Thompson, "A Heuristic Approach to Solving Traveling Salesman Problems," Management Science, 10 (January, 1964), 225-248.

73. E.S. Schwartz, "A Heuristic Procedure for Parallel Sequencing with Choice of Machines," Management Science, 10 (July, 1964), 767-777.

74. E.S. Schwartz, "An Automated Sequencing Procedure with Application to Parallel Programming," Journal of the Association for Computing Machinery, 8 (1961), 513-537.

75. W.J. Gavett, "Three Heuristic Rules for Sequencing Jobs to a Single Production Facility," Management Science, 11 (June, 1965), B166-B176.

76. Little et al., "An Algorithm for the Traveling Salesman Problem," Operations Research, 11 (November, 1963), 972-989.

77. See, for example, Alan J. Rowe, "Toward a Theory of Scheduling," Journal of Industrial Engineering, 11 (March, 1960), 125-136.

78. Baker and Dzielinski, "Simulation of a Simplified Job Shop," Management Science, 6 (1960), 311-323.

79. In Chapter 7, we deal with the "flowshop" problem in which machine sequences for all jobs are identical.

80. See Rowe, "Toward a Theory of Scheduling."

81.J.G. March and H.A. Simon, Organizations (New York: Wiley, 1964), pp. 140-141.

CHAPTER II

FUNDAMENTAL CONCEPTS AND DEFINITIONS

General Problem Statement

The general job shop problem consists of n jobs, each of which requires processing on one or more of m different facilities. In general, each facility is comprised of one or more single-purpose machines, such as drill presses, lathes, or grinders.[1] The route which any job must take through the facilities is specified in advance, and is dictated, at least to a degree, by inviolable technological considerations.[2] Examples of these include the requirements that a hole be drilled before it is tapped, or that a part be sanded before it is painted.

The time required to process a given operation on a given facility has been estimated, and, for the time being, it is assumed that such estimates are correct and invariable. Furthermore, the job file is assumed to be completely known at t=0, and remains fixed throughout the production period.

Representation of Input Data

Before problem analysis can begin, the input data to the problem must be specified in some systematic fashion. The basic inputs are (a) processing times required on each facility for each job, and (b) ordering of operations within jobs. Although many adequate representation modes exist, the most logical and easily interpreted of these is probably the matrix format. Accordingly, this format will now be discussed within the context of a sample problem.

Facility-ordering matrix. Consider a very simple scheduling problem consisting of two jobs and three facilities each consisting of one machine. Processing times, required on each machine for each job, are as follows:[3]

	MACH 1	MACH 2	MACH 3
JOB 1	3	2	5
JOB 2	4	3	1

Figure 3

Additionally, technological considerations dictate the order of operations required for each job:

	SEQUENCE		
JOB 1	M3	M2	M1
JOB 2	M3	M1	M2

Figure 4

While the above mode of representation is adequate for problems of very small size, the matrix format lends itself particularly well to instances of more realistic dimensions. The facility-ordering matrix is simply another way of displaying the information contained in figure 4:

$$F = \begin{bmatrix} 13 & 12 & 11 \\ 23 & 21 & 22 \end{bmatrix}$$

Figure 5

The first digit of each cell contains job information; hence, row 1 relates to job 1, and row 2 is identified with job 2. The second digit of each cell contains sequence information; job 1 requires processing, first on facility 3, second on facility 2, and third on facility 1. Similarly, job 2 requires processing, first on facility 3, second on facility 1, and third on facility 2.

The facility-ordering matrix can, of course, be adapted to any number of facilities and any number of jobs. For a problem having n jobs and m facilities, the matrix will be n x m.

Processing-time matrix. The processing time matrix is used to display processing times required on each facility by each job. The data of figure 1 is converted as follows:

$$P = \begin{bmatrix} 5 & 2 & 3 \\ 1 & 4 & 3 \end{bmatrix}$$

Figure 6

As in the case of F, row 1 relates to job 1, and row 2 is identified with job 2.

Simultaneous consideration of F and P yields the total input picture: Job 1 is processed, sequentially, on machine 3 for 5 minutes, machine 2 for 2 minutes and machine 1 for 3 minutes. Job 2 is processed, sequentially, on machine 3 for

1 minute, machine 1 for 4 minutes, and machine 2 for 3 minutes.

The processing time matrix can also be adapted to any number of facilities and any number of jobs. Where n jobs and m facilities exist, the matrix will be n x m.

The Gantt Chart and Feasible Schedules

Once the input data has been completely specified in the facility-ordering and processing-time matrices, the next step is to consider the generation of feasible schedules. A feasible schedule is some configuration of operations such that (a) all input specifications are maintained, and (b) no two jobs are processed on the same facility at the same time.

The generation of feasible schedules is more easily accomplished using a technique originally devised by Henry L. Gantt in 1920. This procedure makes use of a simple, graphical device, consisting of a rectangular coordinate system in which time is measured along the horizontal axis, and jobs are designated along the vertical axis.

The following Gantt Chart depicts one feasible schedule for the sample problem:

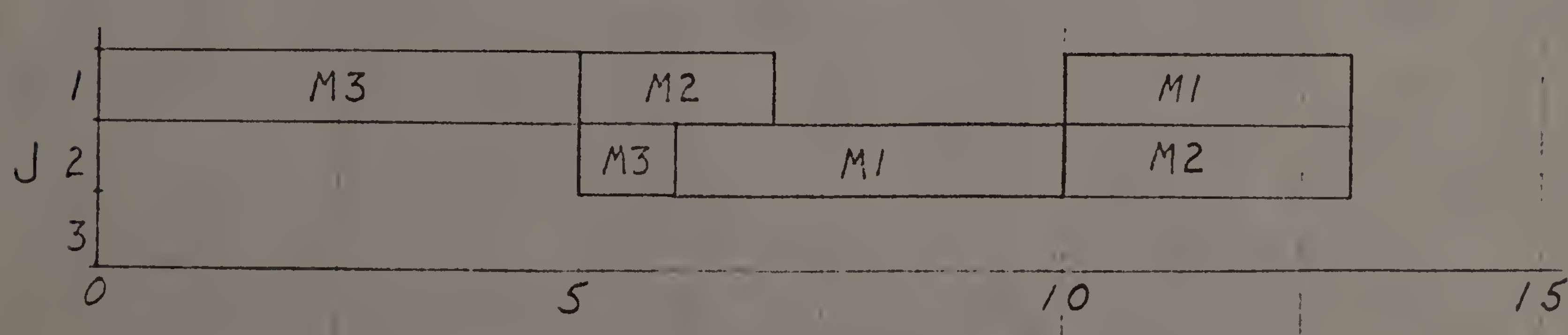


Figure 7

Interpretation is as follows: Job 1 is processed, sequentially, for 5 minutes on machine 3, for 2 minutes on machine 2, and for 3 minutes on machine 1. Job 2 is processed, sequentially, for 1 minute on machine 3, for 4 minutes on machine 1, and for 3 minutes on machine 2. The schedule is feasible because (a) block lengths and sequences are consistent with processing time and facility-ordering specifications, and (b) no two jobs are processed on the same facility at the same time. (The latter can be verified by noting that a perpendicular line, drawn at any point, will not intersect any two blocks bearing the same machine designation.)

The following two Gantt charts do not represent feasible schedules:

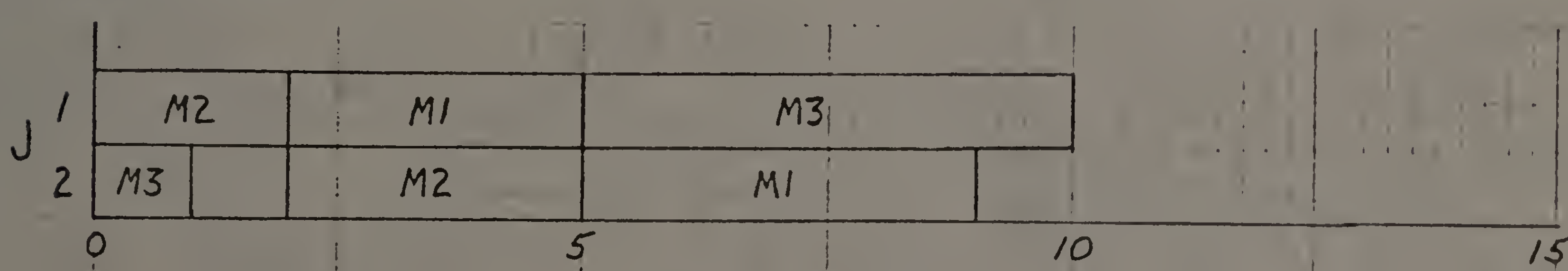


Figure 8

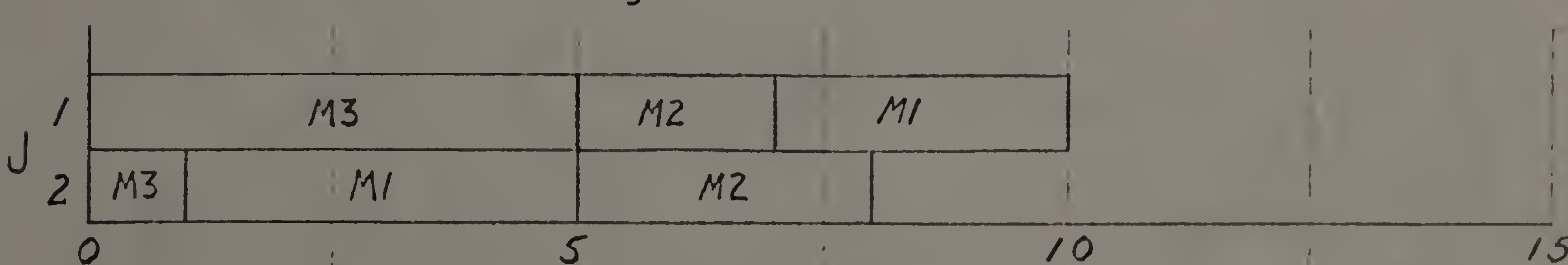


Figure 9

In figure 8, the facility-ordering specification has been violated; in figure 9, both jobs are scheduled to be processed simultaneously on the same facility.

Using the combinatorial formula, it may be determined that there are a total of $2^3=8$ feasible schedules, of which figure 7 is but one example. Two other possibilities are as follows:

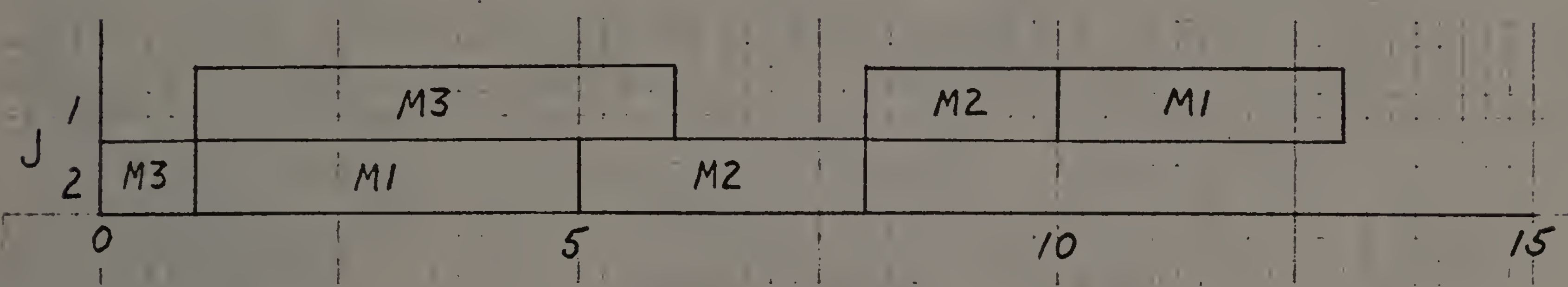


Figure 10

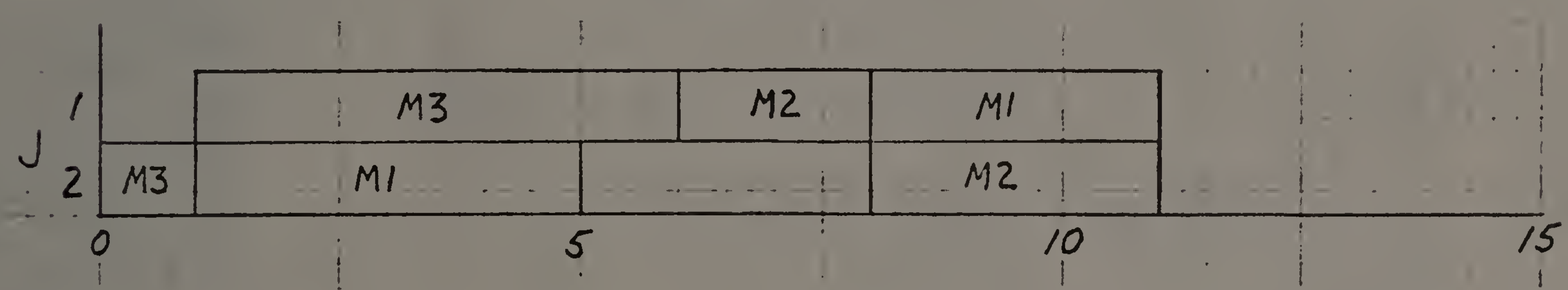


Figure 11

Intuitively, it is clear that not all feasible schedules, in this or any other problem, will be equally desirable. However, before the analyst can amke a choice among schedules, he must first establish some criterion upon which to base his choice.

The Concept of an Optimal Schedule

An <u>optimal schedule</u> is a feasible schedule which optimizes some criterion.[4] While criteria vary from situation to situation, two very commonly encountered ones are makespan (the interval of time from the start of processing until all jobs are completed) and idle time. In each instance, the objective of the schedules is to generate a configuration that will result in minimization of the criterion.

In the problem at hand, the optimal schedule can be determined by exhaustive enumeration, since there are only 8 possibilities. It would be a relatively simple matter to generate all 8 schedules, and then choose the best one on the basis of the selected criterion. Unfortunately, as noted in Chapter 1, the real world is not nearly so elementary an affair.

Satisficing: The Concept of a Satisfactory Schedule

As a practical alternative to seeking optimality, we advocate an intuitive procedure which is deemed likely to produce, at minimum, a marginally acceptable schedule. The method is as follows:

(1) Several schedules are generated, using simple priority rules and supplemental heuristics at each decision point.

(2) Each schedule is then evaluated on the basis of some criterion. The schedule exhibiting the best characteristics is selected and used in the production process.

If the "best" schedule falls short of expectations, the assumption is that it will be used anyway. The decision to employ an imperfect methodology is, in the first place, an admission that alternative modes of analysis are even less desirable. Under these circumstances, there can be little quarrel with the outcome.

The sanctioning of trade-offs is hardly an uncommon management practice. Recall March and Simon's observation, namely, that "most human decision-making, whether individual or organizational, is concerned with the discovery and selection of satisfactory alternatives; only in exceptional cases is it concerned with the discovery and selection of optimal alternatives. To optimize requires processes several orders of magnitude more complex than those required to '*satisfice*'."[5]

There is, therefore, a great deal of justification for choosing the route we have selected. In ensuing chapters, the proposed methodology is fully explained and applied to a problem of realistic dimensions.

Footnotes

1. For purposes of simplicity, the number of machines in each facility is held to one throughout the analysis.

2. Alternate routings are sometimes permitted.

3. We arbitrarily assume times in minutes; the discussion, however, is perfectly general and holds equally well for other units (hours, days, etc.)

4. Multiple criteria are sometimes considered simultaneously.

5. March and Simon, Organizations, pp. 140-141.

C H A P T E R III

THE APPLICATION OF PRIORITY RULES

Basic Definitions

Fundamental to any scheduling problem is the notion of a queue. A queue may be thought of as a "line" of jobs that await processing on a particular facility. The order in which these jobs are actually processed depends on rankings established through the application of certain priority rules. The act of assigning jobs to facilities is known as dispatching. Any facility which exhibits a persistent tendency to form queues is known as a bottleneck facility.

Figure 12 depicts a hypothetical situation in which one job (J1) is currently being processed on machine 5, while two others (J2 and J3) await processing. If certain assumptions

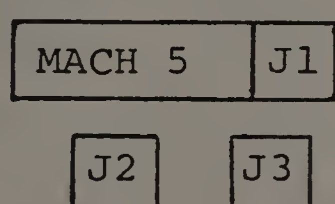


Figure 12

are now made with respect to processing times, the information in figure 12 can be transferred to a Gantt Chart:

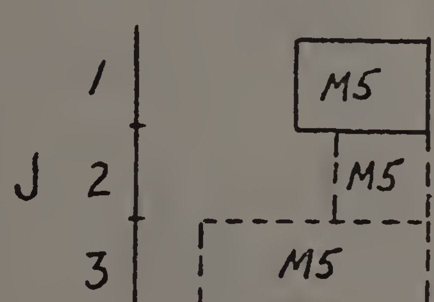


Figure 13

Using either figure as a reference, it is clear that some rule must now be invoked to discriminate between J2 and J3, since only one job can be processed at a time. The next section contains an in-depth discussion of such rules.

Some Basic Priority Rules

We shall illustrate the application of priority rules using a 6x6 scheduling problem defined by the following matrices:

$$P = \begin{bmatrix} 3 & 8 & 8 & 7 & 8 & 6 \\ 4 & 1 & 5 & 4 & 4 & 1 \\ 9 & 8 & 1 & 4 & 1 & 6 \\ 0 & 6 & 8 & 2 & 5 & 4 \\ 6 & 7 & 1 & 1 & 3 & 9 \\ 2 & 8 & 7 & 8 & 3 & 0 \end{bmatrix}$$

Figure 14

$$F = \begin{bmatrix} 14 & 15 & 11 & 16 & 12 & 13 \\ 25 & 23 & 22 & 24 & 26 & 21 \\ 31 & 33 & 35 & 36 & 34 & 32 \\ 42 & 41 & 45 & 44 & 43 & 46 \\ 53 & 56 & 52 & 51 & 55 & 54 \\ 63 & 65 & 64 & 62 & 66 & 61 \end{bmatrix}$$

Figure 15

As before, we make the assumption that each facility contains one machine.

Priority rules not associated with efficiency criteria. The most common of these are the *random* and *earliest arrival rules*. Neither of these takes into account either idle time or makespan, but they do provide a systematic way of generating feasible schedules.

Since there is very little logic to the notion that waiting jobs should be dispatched at random when a facility becomes idle, the random priority rule will not be illustrated.[1] The earliest arrival or "*first-come, first-served*" rule is applied to the sample problem as follows:

Step 1: Construct a temporary, though infeasible, Gantt schedule, by left-justifying all operations on all jobs as much as possible.[2] (See figure 17, tableau 1). At t=0, all machines are idle, and jobs are queued in front of machines as follows:

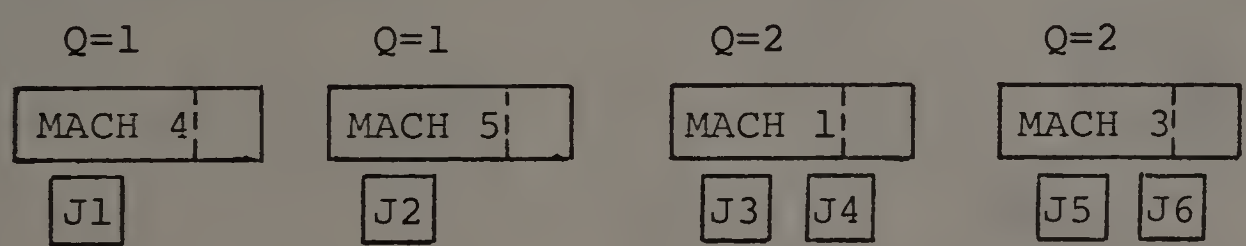
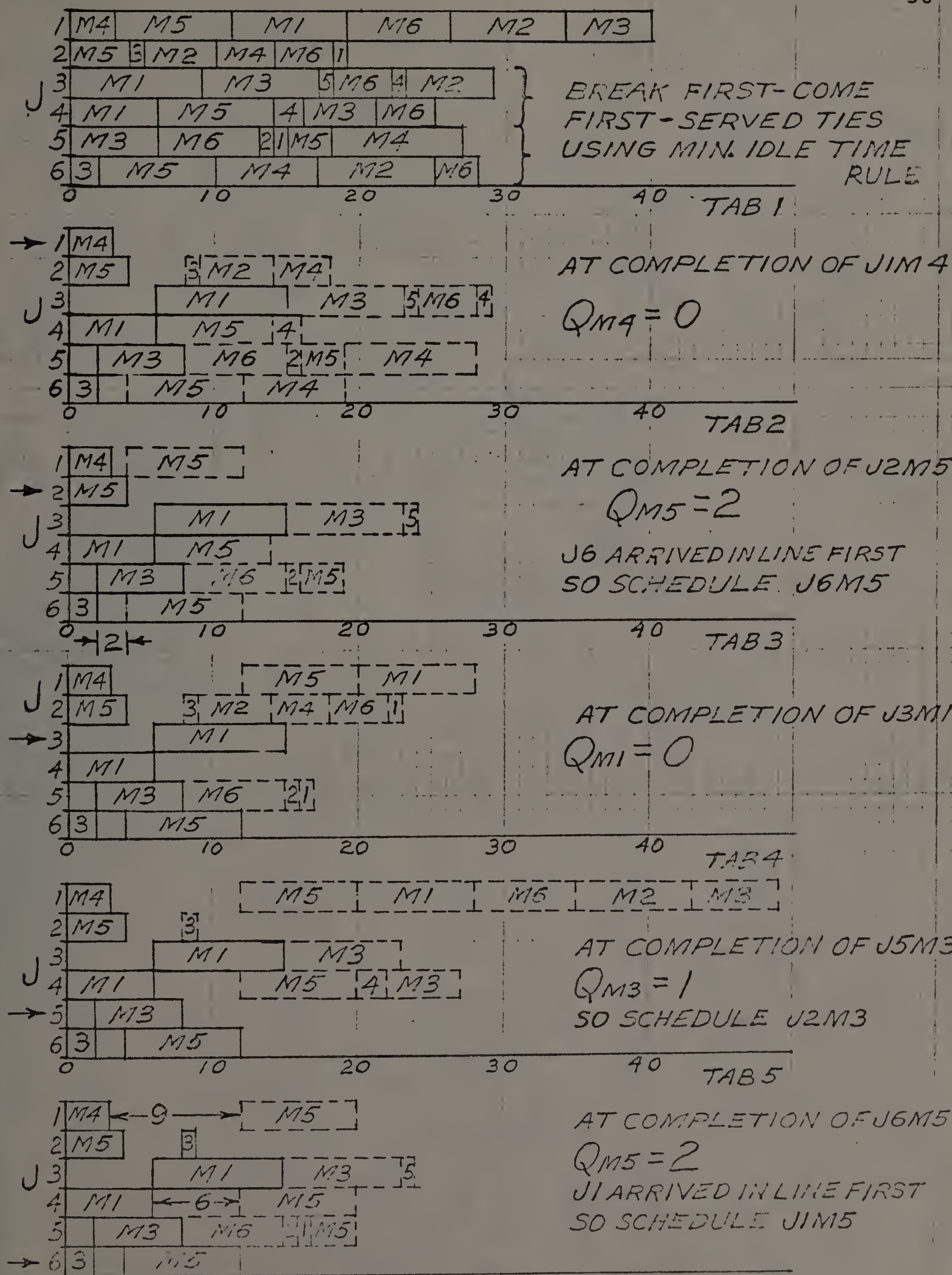


Figure 16

Clearly, no priority rule need be applied in the cases of machines 4 and 5, since only one job is waiting to be processed in each instance. (Priority rules are never applied when the queue length is either 1 or 0.) However, machines 1 and 3 both have queues of length 2.



BREAK FIRST-COME FIRST-SERVED TIES USING MIN. IDLE TIME RULE
TAB 1
AT COMPLETION OF J1M4
$Q_{M4} = 0$
TAB 2
AT COMPLETION OF J2M5
$Q_{M5} = 2$
J6 ARRIVED IN LINE FIRST SO SCHEDULE J6M5
TAB 3
AT COMPLETION OF J3M1
$Q_{M1} = 0$
TAB 4
AT COMPLETION OF J5M3
$Q_{M3} = 1$
SO SCHEDULE J2M3
TAB 5
AT COMPLETION OF J6M5
$Q_{M5} = 2$
J1 ARRIVED IN LINE FIRST SO SCHEDULE J1M5

FIG. 17

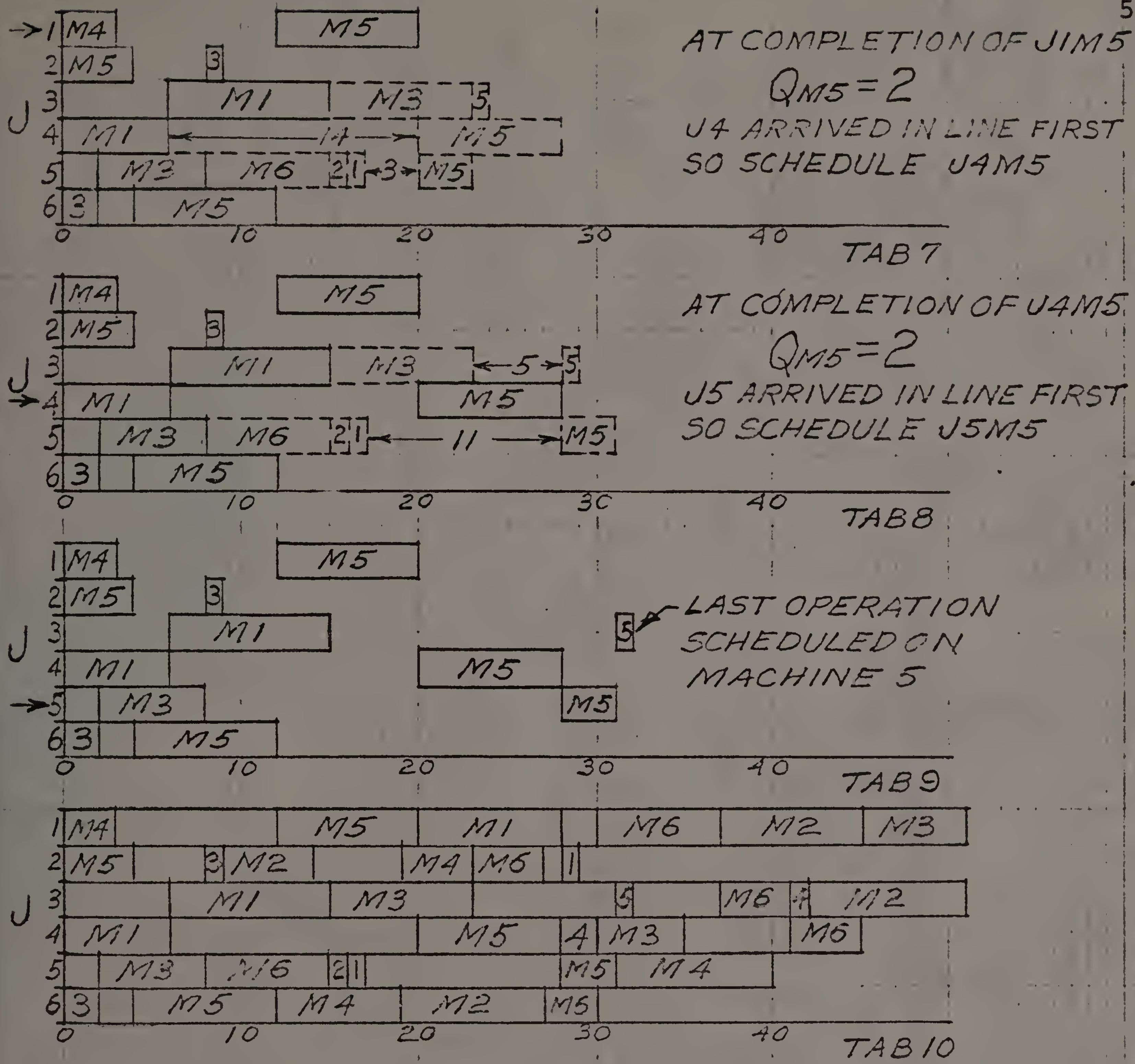

AT COMPLETION OF J1M5
$Q_{M5} = 2$
J4 ARRIVED IN LINE FIRST
SO SCHEDULE J4M5
TAB 7
AT COMPLETION OF J4M5
$Q_{M5} = 2$
J5 ARRIVED IN LINE FIRST
SO SCHEDULE J5M5
TAB 8
LAST OPERATION
SCHEDULED ON
MACHINE 5
TAB 9
TAB 10

FIG. 17 (CONT.)

If the earliest arrival rule is applied to machines 1 and 3, a stalemate results in each case, since all jobs are presumed to have arrived at precisely the same time (t=0). As frequently happens, the stalemate must be broken by temporarily employing some other rule. In this case, we have chosen to dispatch jobs in a way that minimizes the generation of idle time at each decision point.[3] (J4M1 and J6M3 are processed first.)

Step 2. Having permanently scheduled the first operation of each job, repeat the inspection of job rows. Resolve conflicts on those machines having queues of length 2 or longer by assigning priority on the basis of earliest arrival. Repeat the procedure until there are no further conflicts on any machines.

The next scanning of job rows indicates that a queue of length 2 will form in front of machine 5, first during the processing of job 2 (tableau 3), and again during the processing of job 6 (tableau 6). In the first instance, J6 arrives first, so it is scheduled next on machine 5; in the second instance, J1 arrives first, so it is scheduled next, also on machine 5.

The next pass reveals two more conflicts at machine 5. (Tableaus 7 and 8). These are handled by scheduling job 4 first in the former instance, and job 5 first in the latter.

Step 3. Once it has been determined that no further jobs await immediate processing on any facility (Q=0 for all jobs), proceed to schedule the remaining operations around those already scheduled. If, in this process, new queuing problems arise, resolve them in the manner previously described.

Tableau 10 shows the final schedule. Jobs have been scheduled from left to right, taking careful note of machine-sequence requirements. In no instance were waiting-lines greater than length 1; at no time, therefore, was it necessary to invoke the priority rule.

In this example, machine 5 exhibits the characteristics of a bottleneck facility, since it generates queues on a large number of occasions. The analysis of bottleneck facilities is generally very important, since they tend to establish the configuration of the entire schedule. In some cases, it is desirable to predict the existence of bottlenecks, prior to the laying out of schedules; in order to determine this contour. As will be seen, this is accomplished by calculating, in advance, total utilization for each facility.

The basic drawback of the earliest arrival method is that it is not based on a meaningful job-shop criterion. In the case of human queues, where waiting time is the crucial issue, the servicing of patrons according to the earliest arrival criterion is dictated by the precepts of fairness. The situation is quite different in the case of jobs, however, which are not nearly so apt to become impatient!

In job-shops, it is the customers who must be kept satisfied, and not the jobs themselves. To this end, managers are primarily interested in finishing jobs as soon as possible and are only secondarily concerned with the order in which they are started.

In the sections that follow, we discuss three rules that are much more consistent with the logical goals of shop managers.

Priority rules for compact schedules. By our definition, a compact schedule is one that exhibits a small amount of idle time and/or a short makespan. Schedules of this type are likely to permit the processing of jobs within acceptable time periods.

Shortest imminent operation rule. The shortest imminent operation rule dictates that queuing conflicts be resolved by giving first priority to the job with the shortest impending operation. This is tantamount to scheduling operations in a manner that minimizes the generation of idle time.

Application of the rule proceeds as follows:

Step 1. Construct a temporary, though infeasible, schedule by left-justifying all operations on all jobs as much as possible. For the problem at hand, this results in the schedule of figure 18, tableau 1.

Step 2. Permanently schedule the first operation of each job by considering the job rows in sequence. Resolve

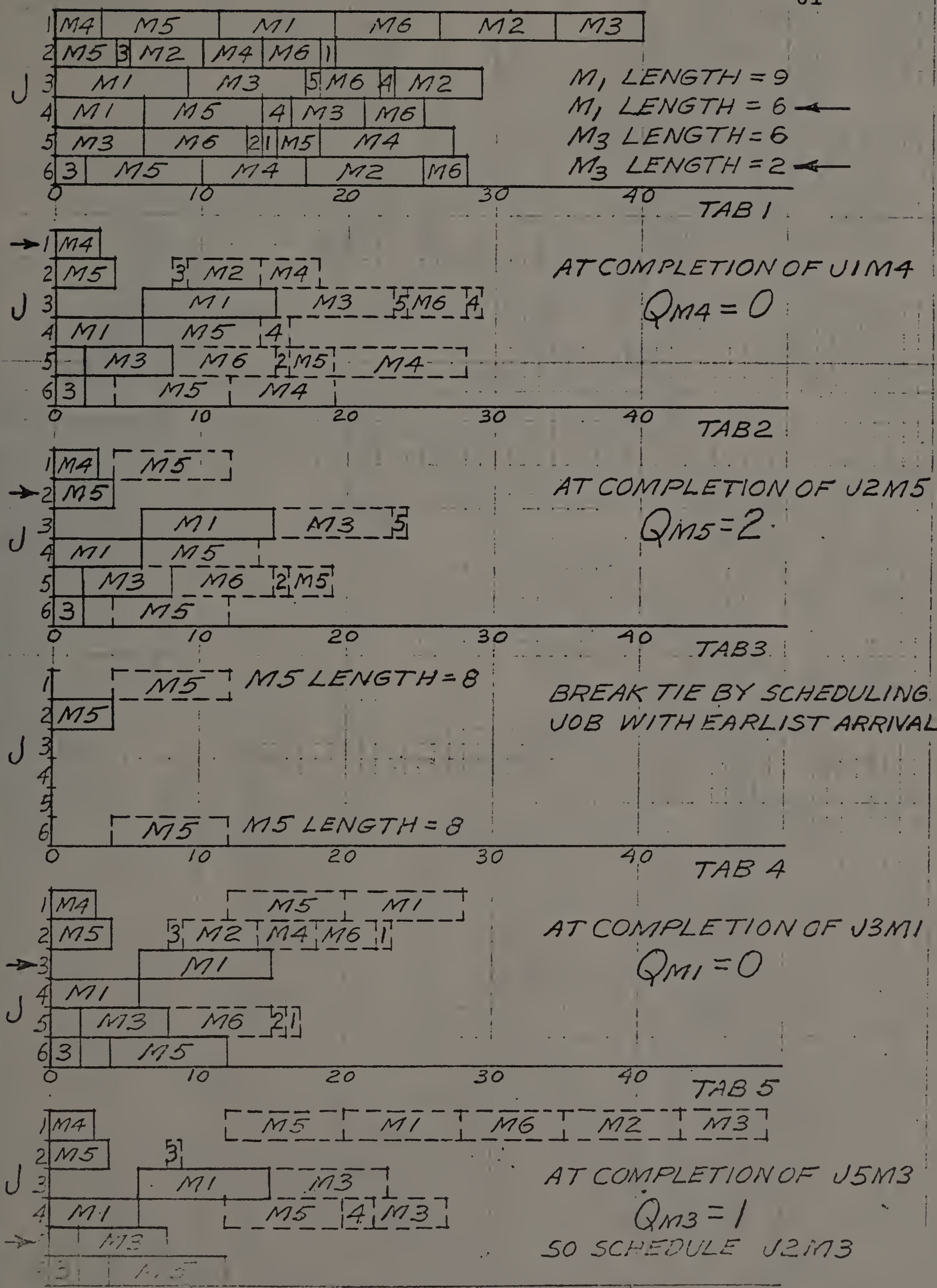

M, LENGTH = 9
M, LENGTH = 6
M3 LENGTH = 6
M3 LENGTH = 2
TAB 1
AT COMPLETION OF J1M4
QM4 = 0
TAB2
AT COMPLETION OF J2M5
QM5 = 2
TAB3
M5 LENGTH = 8
BREAK TIE BY SCHEDULING JOB WITH EARLIST ARRIVAL
M5 LENGTH = 8
TAB 4
AT COMPLETION OF J3M1
QM1 = 0
TAB 5
AT COMPLETION OF J5M3
QM3 = 1
SO SCHEDULE J2M3
TAB 6

FIG. 16

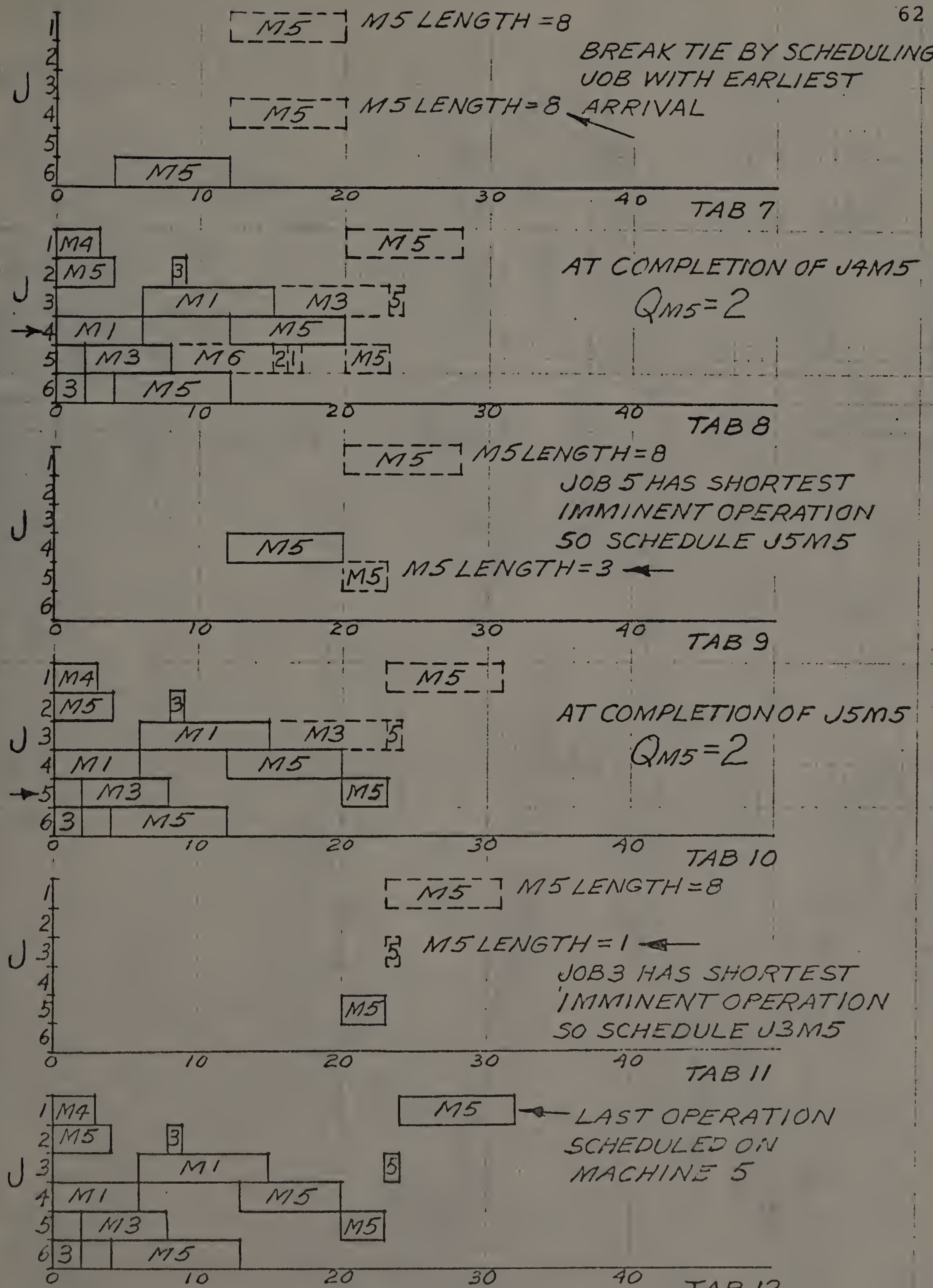

M5 LENGTH = 8
BREAK TIE BY SCHEDULING JOB WITH EARLIEST ARRIVAL
M5 LENGTH = 8
J
TAB 7
AT COMPLETION OF J4M5
$Q_{M5} = 2$
TAB 8
M5 LENGTH = 8
JOB 5 HAS SHORTEST IMMINENT OPERATION SO SCHEDULE J5M5
M5 LENGTH = 3
TAB 9
AT COMPLETION OF J5M5
$Q_{M5} = 2$
TAB 10
M5 LENGTH = 8
M5 LENGTH = 1
JOB 3 HAS SHORTEST IMMINENT OPERATION SO SCHEDULE J3M5
TAB 11
LAST OPERATION SCHEDULED ON MACHINE 5
TAB 12
M1
M3
M4
M5
M6
0
10
20
30
40

FIG. 18 (CONT)

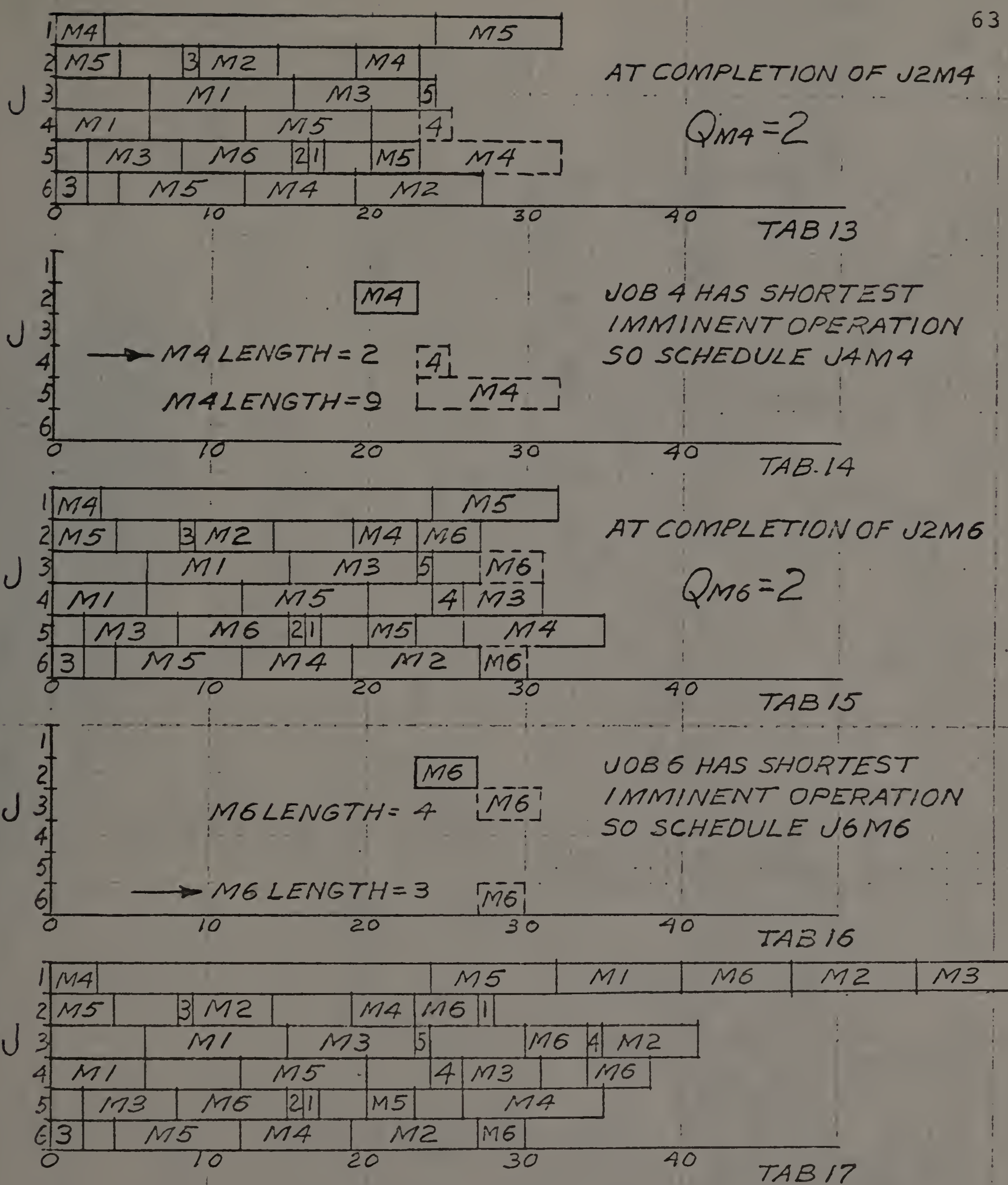

AT COMPLETION OF J2M4
QM4=2
TAB 13
M4 LENGTH = 2
M4 LENGTH = 9
JOB 4 HAS SHORTEST
IMMINENT OPERATION
SO SCHEDULE J4M4
TAB.14
AT COMPLETION OF J2M6
QM6=2
TAB 15
M6 LENGTH = 4
M6 LENGTH = 3
JOB 6 HAS SHORTEST
IMMINENT OPERATION
SO SCHEDULE J6M6
TAB 16
TAB 17

FIG 18(CONT)

conflicts on those machines having queues of length 2 or longer by assigning priority on the basis of shortest operation.

In tableau 1, queues of length 2 are seen to have formed in front of machines 1 and 3 at t=0. These conflicts are resolved by dispatching the shortest operation in each case, namely J4M1 and J6M3.

Step 3. Continue the sequential examination of job rows, resolving conflicts as they occur, until no further conflicts are seen to exist on any machines. Then, schedule the remaining operations around those already scheduled. If new queuing problems arise, resolve them in the manner previously described.

In tableaus 3 and 6, queuing problems arise with respect to machine 5. Attempts at applying the shortest imminent operation rule fail because both imminent operations are of the same length. Hence, it is necessary to invoke an alternate rule. (The earliest arrival rule is applied in tableaus 4 and 7.)

Conflicts arising in tableaus 8 and 10 are handled routinely, the shortest imminent operation rule being readily applicable in each case. Since all conflicts are exhausted at tableau 10, scheduling of remaining operations begins in tableau 11. Further queuing difficulties are encountered in the scheduling of M4 (tableau 13) and later in the scheduling of M6 (tableau 15). Both of these are resolved routinely (tableaus 14 and 16). The final schedule is exhibited in tableau 17.

Since idle time was of implicit concern throughout the scheduling process, the ultimate schedule would be expected to rank high in terms of "compactness." Since compactness was of no concern in the generation of the "first-come, first-served" schedule (figure 17, tableau 10), a comparison of the two would be in order:

	FC/FS	SIO
JOB 1	51	61
JOB 2	29	28
JOB 3	51	41
JOB 4	45	38
JOB 5	40	35
JOB 6	30	30
TOTAL	246	233

Figure 19

JOB PROCESSING TIMES UNDER EARLIEST ARRIVAL AND SHORTEST IMMINENT OPERATION RULES (SAMPLE PROBLEM).

On the basis of figure 19, we observe that the shortest imminent operation rule produced shorter processing times than the earliest arrival rule for four out of six jobs. (In the case of job 6, the processing times were the same.) Also, the total non-elapsed time required for the processing of all jobs on all machines was considerably shorter when the earliest arrival rule was employed.

Recognizing the futility in attempting to draw elaborate inferences from a sample of one, we proceed to the next rule.

Bottlenecks-first rule. In scheduling problems, inspection of machine utilization data will reveal the extent to which demands are placed on any given facility. A very persuasive line of reasoning suggests that first priority in scheduling should be given to those facilities on which jobs require processing for comparatively long periods of time (i.e. potential bottleneck facilities).

The bottlenecks-first rule is based on the premise that total utilization of the longest-utilized facility represents a lower limit on makespan for any given set of jobs.[4] To illustrate this fundamental point, consider total utilization for each machine in the sample problem:

M1	25
M2	28
M3	28
M4	26
M5	32
M6	29

Figure 20

Without examining the situation further, it is clear that under no circumstances could makespan be less than 32 time units, the sum of all operational times on machine 5. When all other problem parameters (lengths of individual operations, machine sequences) have been considered, a makespan of 32 will probably not be achievable; nevertheless, it is a valid measure of the length of the best possible schedule, in terms of makespan, under the best possible circumstances.

The bottlenecks-first rule dictates that facilities be scheduled in order of their total utilization. The first stage of the procedure involves construction of a preliminary schedule, containing only operations on the most utilized facility. Following this, the second-most utilized facility is scheduled, consistent both with prior scheduling and with the machine-sequence requirements of jobs. Scheduling continues in this fashion until all facilities are exhausted.

A summary of procedural rules, with application to the sample problem, is now given.

Step 1. Determine total utilization for each machine, and establish a utilization ranking. For the problem at hand (see figure 20), the utilization ranking (most utilized first) is either 5 6 2 3 4 1 or 5 6 3 2 4 1.

Step 2. Rearrange the data of the facility-ordering matrix to show the approximate sequence in which jobs require the use of any given machine. For the problem under consideration, this would be done as follows:

	OPN 1	OPN 2	OPN 3	OPN 4	OPN 5	OPN 6
MACH 1	3	4	1	5	--	2,6
MACH 2	4	--	2,5	6	1	3
MACH 3	5,6	2,3	--	--	4	1
MACH 4	1	--	6	2,4	3	5
MACH 5	2	1,6	3,4	--	5	--
MACH 6	--	5	--	1,3	2,6	4

JOB NUMBERS

Figure 21

As an illustration, the data in the machine 1 row is obtained as follows: Successive examination of the columns of the facility-ordering matrix reveals that machine 1 is used for job 3's first operation, job 4's second operation, job 1's third operation, job 5's fourth operation, job 2's sixth operation and job 6's sixth operation. The data in other rows is obtained in an analogous manner.

Step 3. Construct m schedules (one for each facility) using the m rows of data in the table described in step 2. For each schedule, left-justify all operations as much as possible, consistent with no overlap. In instances where two or more jobs vie for the same sequence position, order them arbitrarily.

For the problem at hand, the resulting schedules are shown in figure 22.

Step 4. Construct a final schedule by transferring information from the m schedules, one at a time, according to

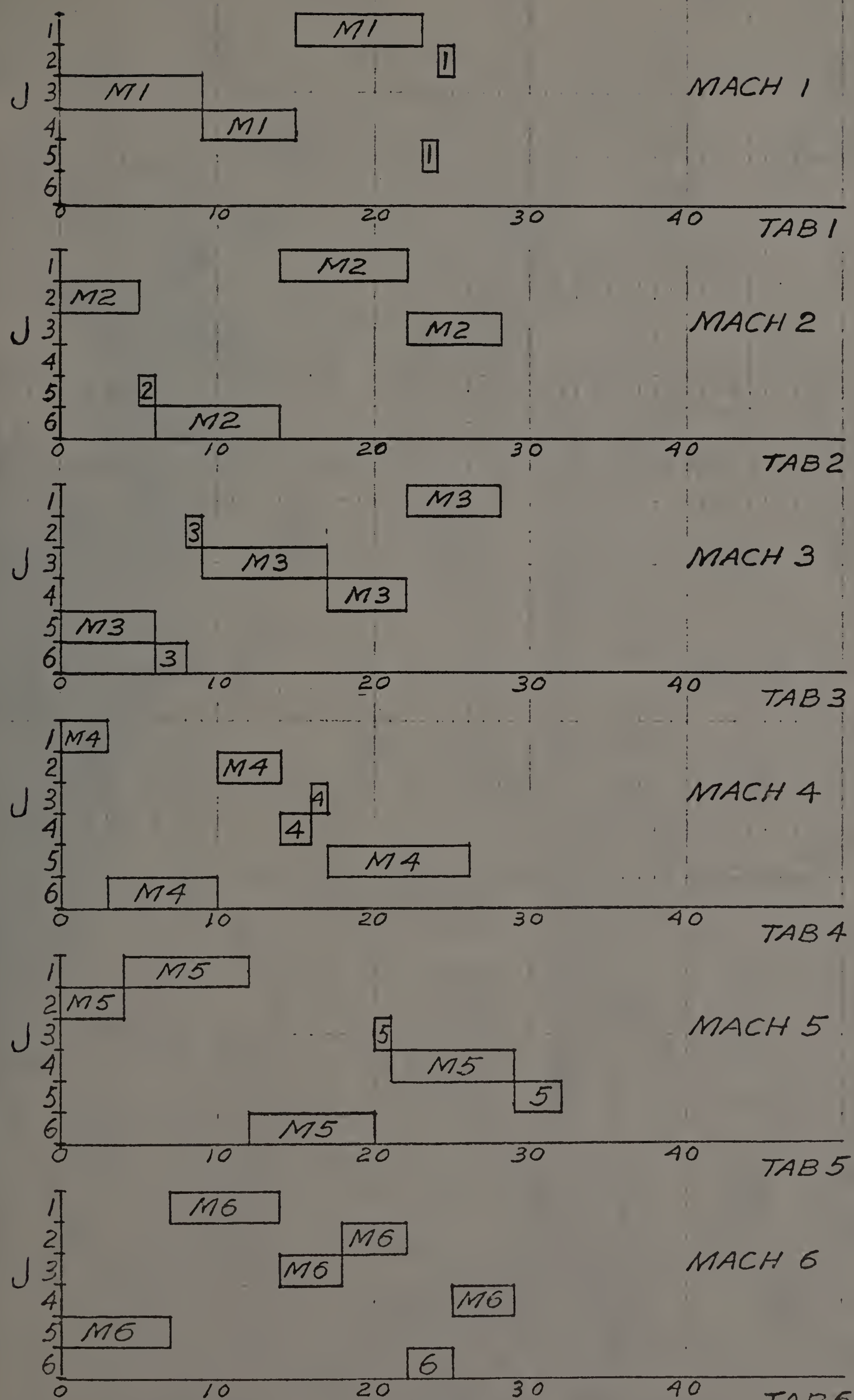

MACH 1
TAB 1
MACH 2
TAB 2
MACH 3
TAB 3
MACH 4
TAB 4
MACH 5
TAB 5
MACH 6
TAB 6
J
0
10
20
30
40

FIG. 22

the order prescribed by the machine-utilization ranking.

To illustrate, we arbitrarily select the machine utilization ranking 5 6 3 2 4 1 for consideration. The schedule for machine 5 (figure 22, tableau 5) is therefore considered first, and the information is transferred directly to tableau 1 of figure 23. Next, the schedule for machine 6 is considered.

In scheduling the operations of machine 6 "around" those of 5, care must be taken to ascertain that the machine-sequence requirements of each job are not violated. Also, wherever possible, provisions must be made for "insertion" of operations on machines yet to be considered.

These highly important points warrant illustration. Using figure 22 as a reference, it is obvious that under no circumstances should J5M6 be scheduled to start earlier than t=6, since J5M3, requiring 6 time units, must precede it. J5M6 is therefore scheduled accordingly in figure 23, tableau 2.

The next operation to be considered is J1M6. (Operations on any given machine are always considered from left to right.) If there were no machine sequencing requirements, J1M6 would be scheduled to begin at t=13. However, J1M1 must directly follow J1M5, and a gap of 8 units must therefore be provided for its insertion at a later time. Thus, J1M6 is scheduled to begin at t=20.

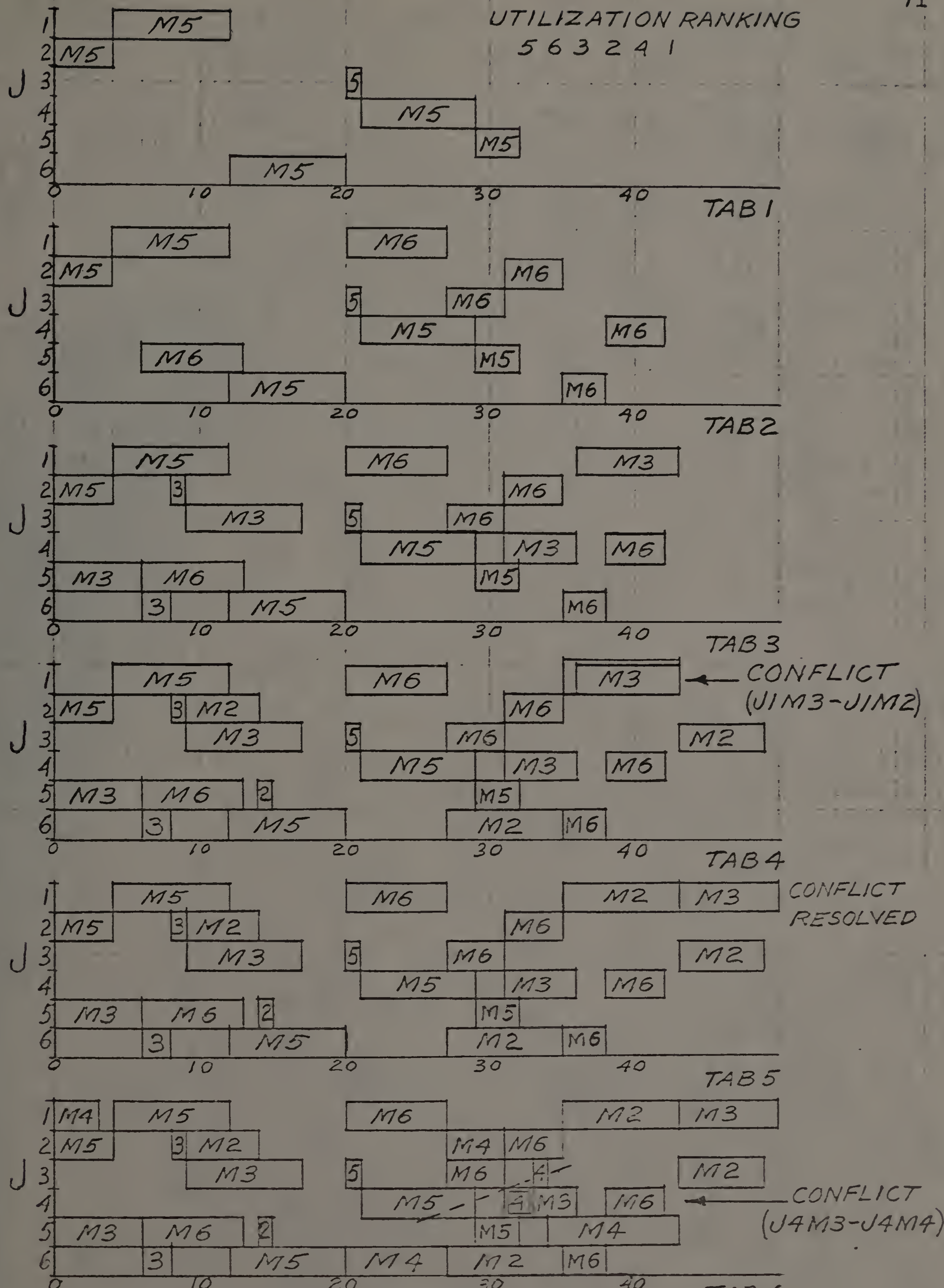
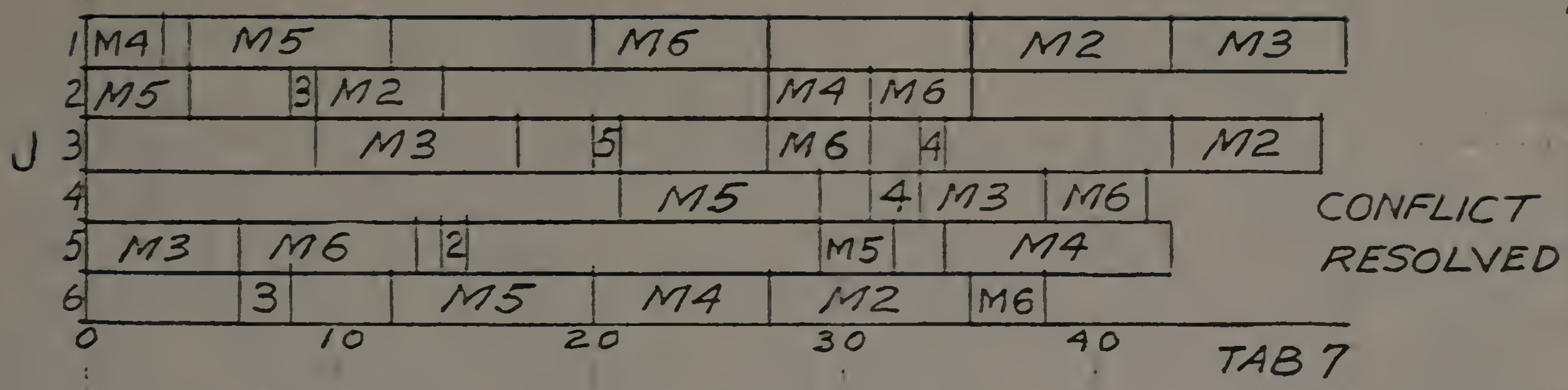
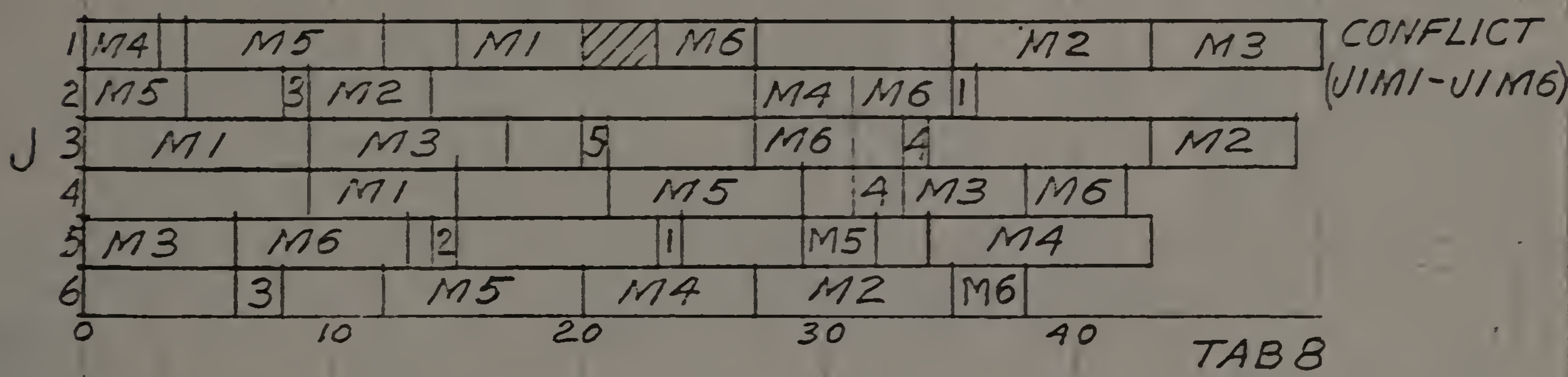

UTILIZATION RANKING
5 6 3 2 4 1
J
TAB 1
TAB 2
TAB 3
TAB 4
CONFLICT
(J1M3-J1M2)
TAB 5
CONFLICT
RESOLVED
TAB 6
CONFLICT
(J4M3-J4M4)

FIG.23


J
1 M4 M5 M6 M2 M3
2 M5 3 M2 M4 M6
3 M3 5 M6 4 M2
4 M5 4 M3 M6
5 M3 M6 2 M5 M4
6 3 M5 M4 M2 M6
0 10 20 30 40
CONFLICT RESOLVED
TAB 7


J
1 M4 M5 M1 M6 M2 M3
2 M5 3 M2 M4 M6 1
3 M1 M3 5 M6 4 M2
4 M1 M5 4 M3 M6
5 M3 M6 2 1 M5 M4
6 3 M5 M4 M2 M6
0 10 20 30 40
CONFLICT (J1M1-J1M6)
TAB 8

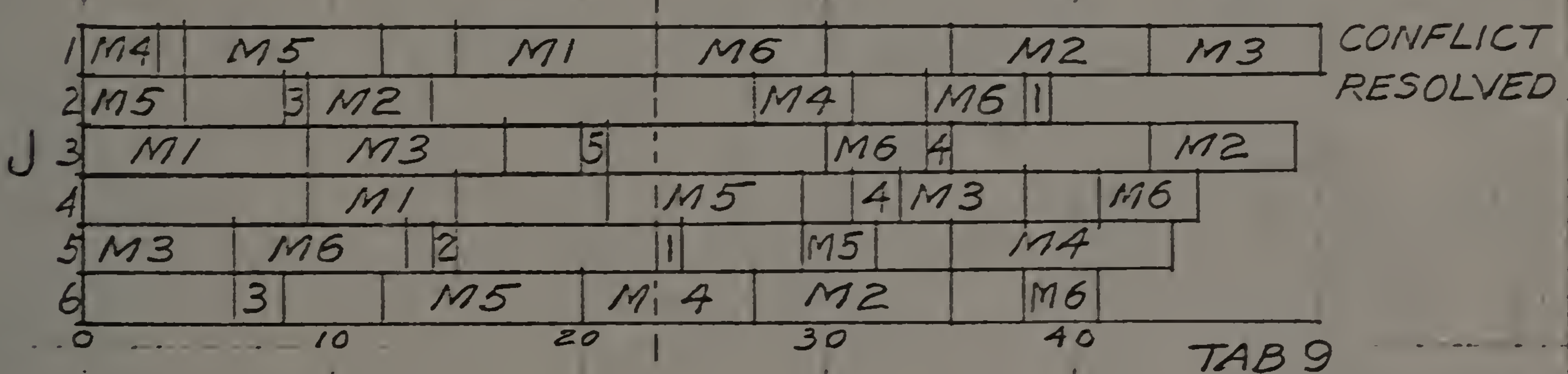

J
1 M4 M5 M1 M6 M2 M3
2 M5 3 M2 M4 M6 1
3 M1 M3 5 M6 4 M2
4 M1 M5 4 M3 M6
5 M3 M6 2 1 M5 M4
6 3 M5 M 4 M2 M6
0 10 20 30 40
CONFLICT RESOLVED.
TAB 9

FIG. 23 (CONT)

As schedule construction proceeds, conflicts may be encountered, due to insufficient space having been left for the insertion of operations. Normally, these conflicts can be resolved (with little detriment to schedule efficiency) by a simple right-shifting of operations, sufficient to accommodate the required insertions.

The following examples are deemed typical:

(1) In the scheduling of M2 (figure 23, tableau 4) a conflict occurs between J1M2 and previously scheduled J1M3. The most obvious way to resolve this conflict is to move J1M3 to the right by 7 units.

(2) In the scheduling of M4 (figure 23, tableau 6), a conflict occurs between J4M4 and previously scheduled J4M3. This is most easily resolved by moving J4M3 to the right by 2 units.

(3) In the scheduling of M1, the conflict between J1M1 and J1M6 is most efficiently resolved by shifting J1M6, J3M6, J2M6, J6M6, and J4M6 to the right by 3 units; by shifting J3M4 and J5M4 to the right by 1 unit; and by shifting J2M1 to the right by 3 units.

The ultimate schedule (figure 23, tableau 9) has makespan of 50.[5] Not surprisingly, this figure is marginally less than that of the "first-come, first served" schedule (51), and considerably less than that of the shortest imminent operation configuration (61).

Although makespan is relatively short in comparison to other schedules, it is still considerably greater than the lower limit of 32. This is because total machine utilization is approximately the same for all machines. (There is only a 7 point difference between machine 5, the most utilized machine, and machine 1, the least utilized.)

Clearly, the concept of scheduling the most utilized machines first has greatest appeal in situations where all machine loadings are not uniformly heavy. In many instances, there will be one or two machines with very high total utilizations, while the remaining machines will be used much less extensively. Such conditions greatly increase the likelihood that makespan can be made to approach its lower limit.

To illustrate this point, consider an extreme case in which the processing time matrix of the sample problem is modified as follows:

$$P = \begin{bmatrix} 5 & 1 & 2 & 7 & 4 & 2 \\ 5 & 2 & 5 & 1 & 4 & 3 \\ 1 & 1 & 3 & 4 & 2 & 1 \\ 1 & 2 & 1 & 3 & 1 & 4 \\ 3 & 7 & 4 & 5 & 1 & 2 \\ 4 & 1 & 2 & 2 & 3 & 2 \end{bmatrix}$$

Figure 24

If the facility-ordering matrix remains unchanged, the application of step 3 results in the schedules shown in figure 25. Note that the problem has been designed to give machine 6 a total utilization approximately twice as great as that of any other.

The machine utilization ranking is now 6 2 1 4 3 5. Based on this, construction of the ultimate schedule is depicted in figure 26. The ultimate schedule (tableau 4) has a makespan of 32, only three greater than the lower limit of 29.

To sum up, the likelihood that makespan can be made to approach its lower limit varies with the range of the total machine utilization figures. In instances where this range is very wide, the foregoing method would be expected to yield especially good results.

In practice, utilization of the "bottlenecks-first" rule is very common, because of the frequency with which large, diverse shop operations are encountered. Since large numbers of jobs and machines contribute to a very wide range of machine utilization figures, the likelihood that makespan can be made to approach its lower limit is markedly increased. Thus, the "bottlenecks first" rule is recognized as yielding especially good results.

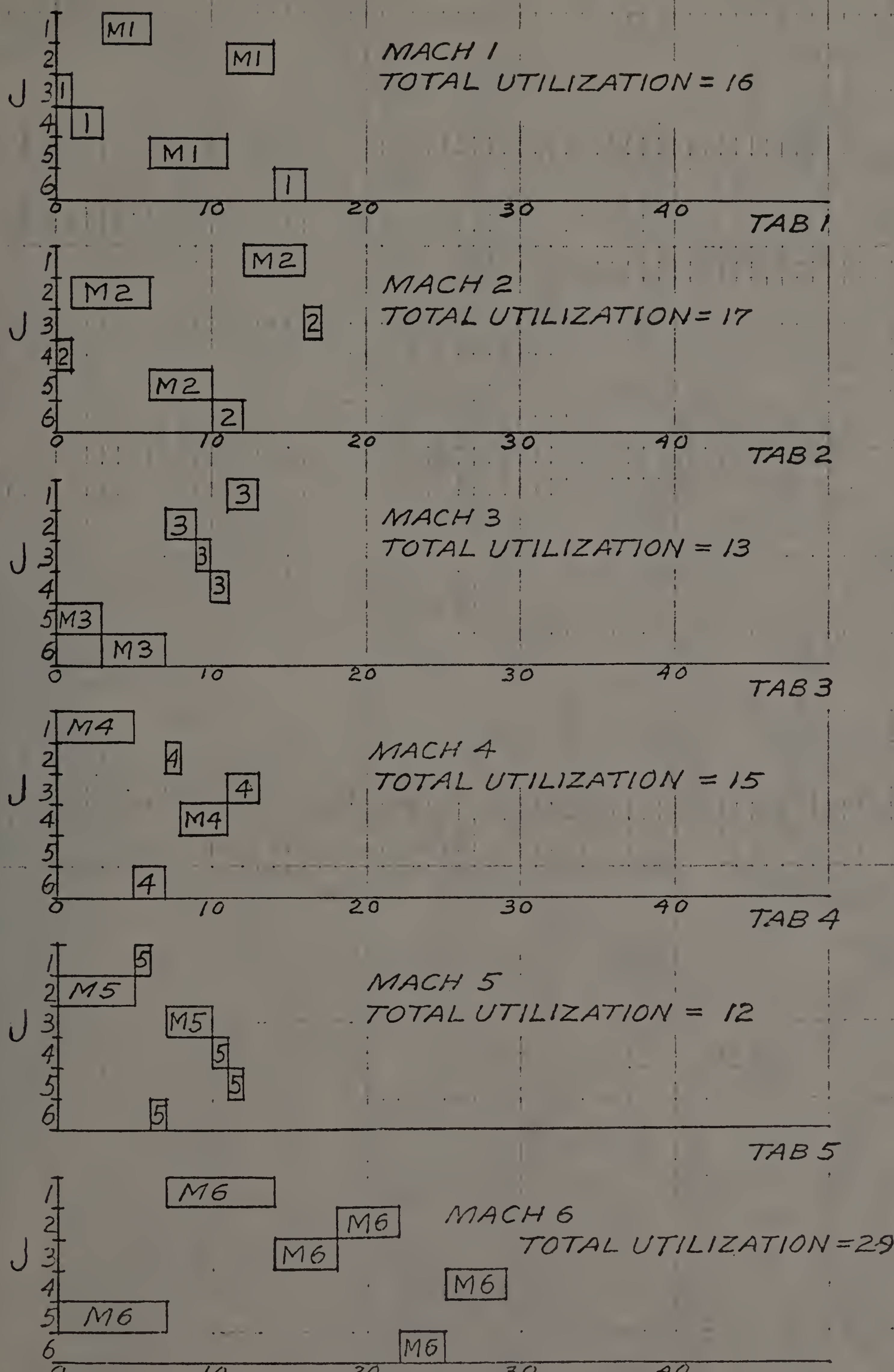

MACH 1
TOTAL UTILIZATION = 16
TAB 1
MACH 2
TOTAL UTILIZATION = 17
TAB 2
MACH 3
TOTAL UTILIZATION = 13
TAB 3
MACH 4
TOTAL UTILIZATION = 15
TAB 4
MACH 5
TOTAL UTILIZATION = 12
TAB 5
MACH 6
TOTAL UTILIZATION =29
TAB 6
J
0
10
20
30
40

FIG.25

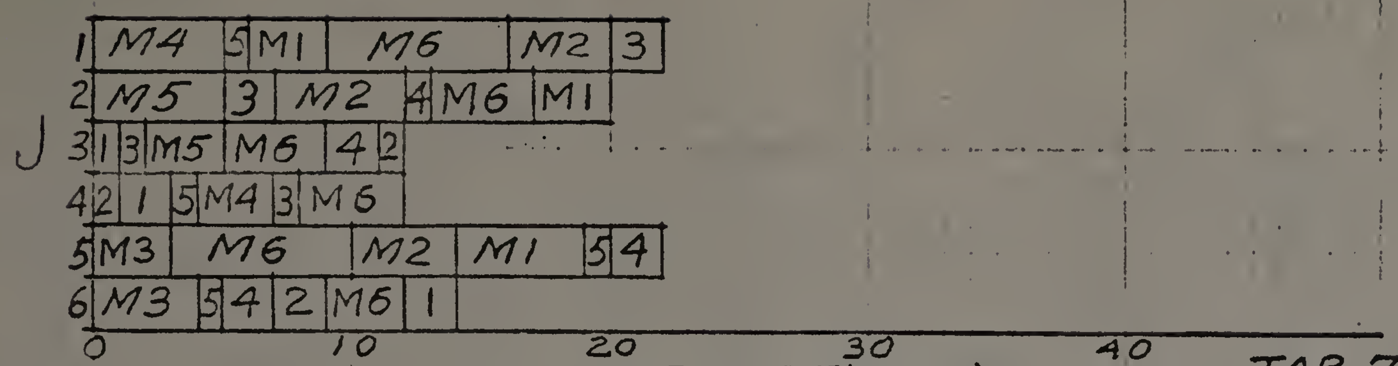


FIG 25(CONT)

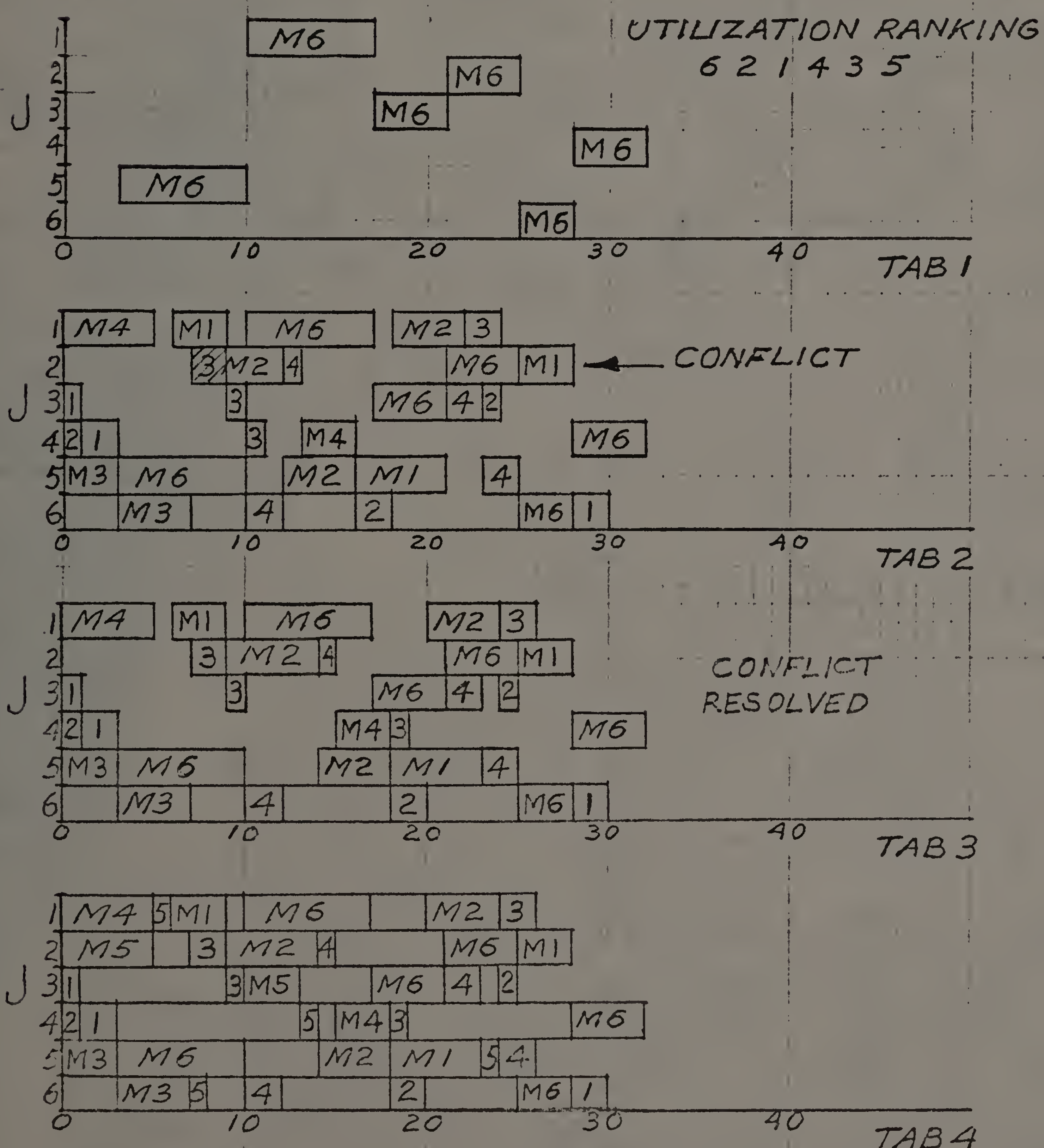


FIG 26

Footnotes

1. The random priority rule, while seldom used in job shops, is sometimes invoked by the phone company. When many calls compete for the use of limited facilities and are therefore delayed, the technique is to pick at random from the "queue," thus equalizing the average waiting time. Callers "at the head" of the queue experience longer waits than if a "first-come, first-served" rule were invoked; on the other hand, however, those last "in line" experience shorter waiting times. The net effect is to reduce the likelihood that any customer will become displeased to the point of outrage. (Note that the system works only because customers are not aware of their positions "in line" and therefore cannot feel cheated by anything short of a first-come, first-served approach.)

2. This is generally the first step in applying any of the rules. Its purpose is simply to display all the data in a form that can be readily assimilated.

3. It is important to recognize that basic priority rules, used independently of heuristics, do not have "look-ahead" features. Thus, for the time being, interest is confined only to the immediate generation of idle time.

4. The reader is reminded that makespan is total elapsed time to completion of the job set.

5. Scheduling according to the alternative utilization ranking 5 6 2 3 4 1 yields identical results.

C H A P T E R IV

DUE-DATE SCHEDULING

Introduction

In the preceding sections, we devoted our attention to the formulation of certain priority rules, the application of which was deemed likely to result in the generation of "compact" schedules. Compactness was defined in terms of either idle time or makespan, such measures being used to assess the efficiency of schedules, with no specific constraints being applied to individual jobs.

In practice, it is frequently necessary to consider due-dates in the scheduling process. A schedule which is judged exemplary on the basis of the criteria of the foregoing sections may be wholly inadequate when due-dates are brought into the picture. The violation of deadlines may turn out to be extremely costly in terms of specific monetary penalties, cancelled orders, lost future sales, and general ill will.

To illustrate these concepts, consider once again the schedule of figure 23, tableau 9, now modified to include due dates (figure 27). The schedule has now become considerably less attractive, since five of the six jobs are completed past their due-dates.

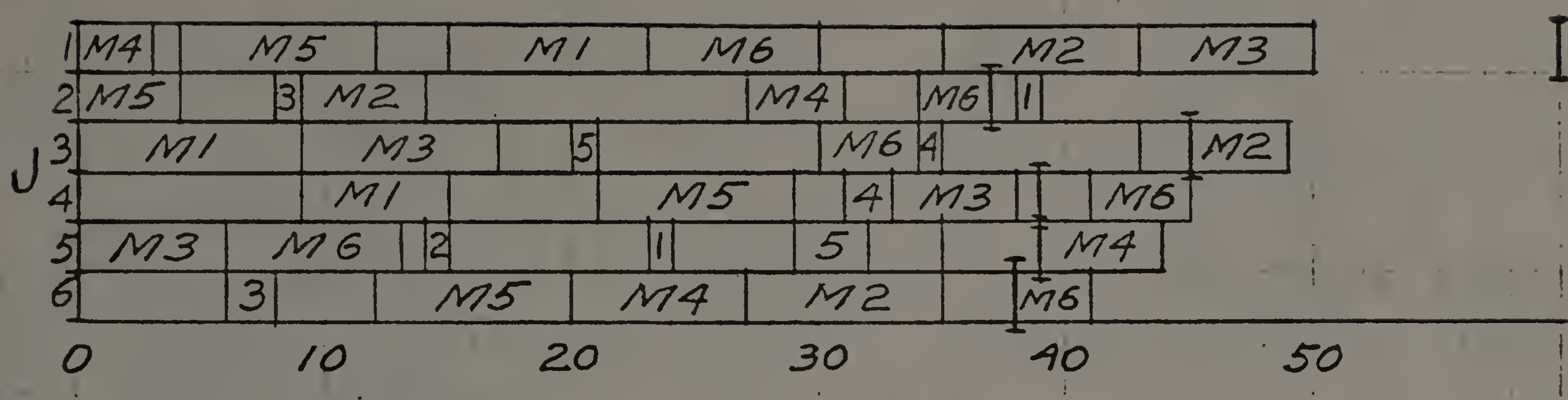


Figure 27

While five of the six jobs are completed late, the last one is completed early. Thus, there exists the possibility of an adjustment that will "stretch out" the early job and "compress" the late ones in such a way that all deadlines are met. This equalizing procedure may not always be totally successful, but improvements can generally be achieved. Barring the meeting of all deadlines, schedules may be judged on the basis of "minimum total lateness" or some similar compromise measure.

If deadlines exist, jobs are not normally accepted unless prospects for on-time completion appear favorable. Risk-oriented managers sometimes accept jobs even if preliminary analysis shows that lateness will occur, but, in general, this is bad practice. The possibility that favorable happenstance will permit perfect accommodation of an otherwise troublesome set of jobs is likely to be far outweighed by the possibility of machine breakdowns, work stoppages and the like.

In scheduling to meet due dates, dispatching decisions are made on the basis of certain rules that explicitly consider impending deadlines. Several of these will now be examined with reference to the sample problem.

Job Slack Rule

The fundamental job slack rule asserts that priority during dispatching should always be given to those jobs which have the nearest deadlines, hence the greatest likelihood of coming in late. To illustrate, consider the sample problem, with due dates as shown below:

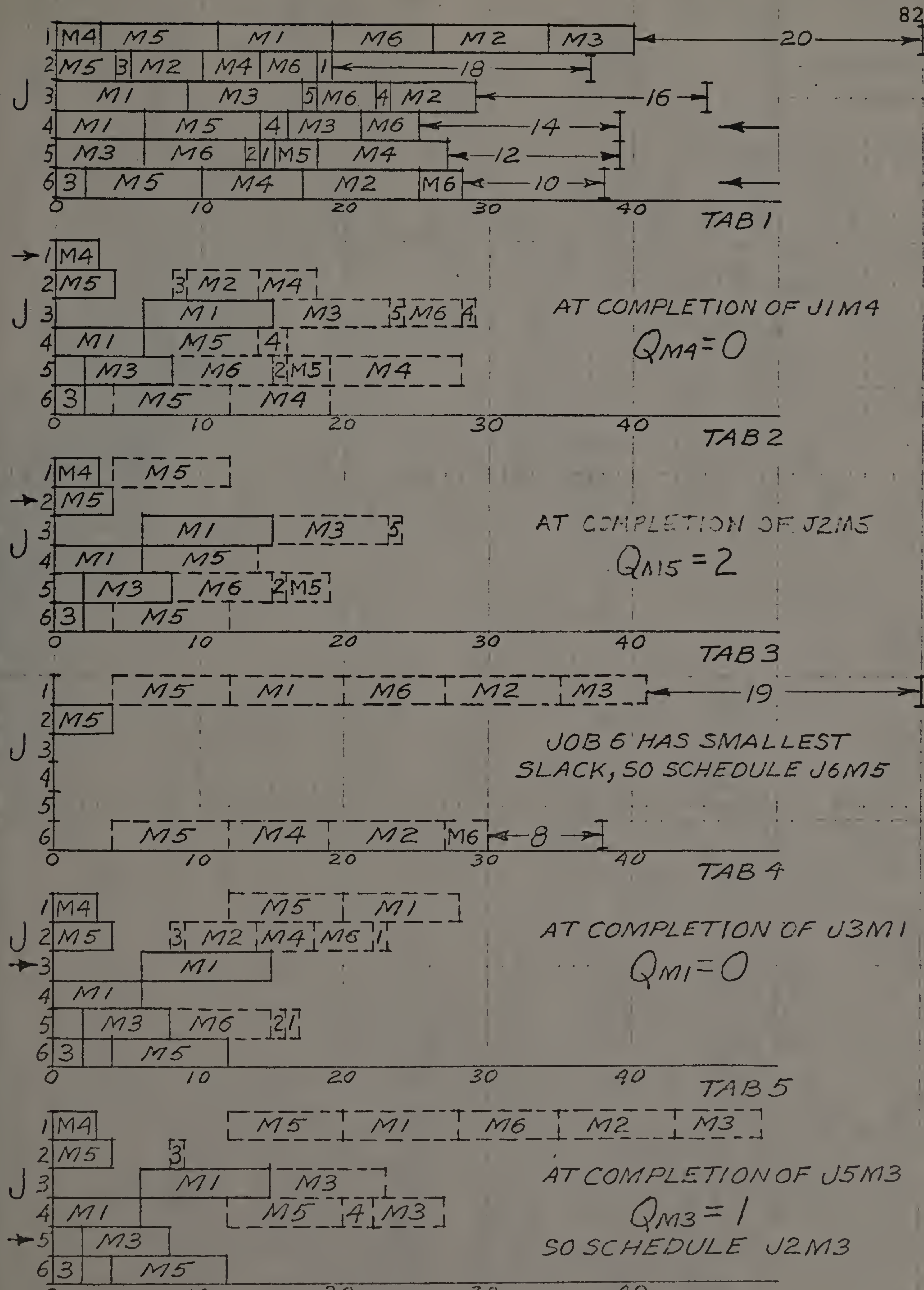
JOB	DUE DATE
1	t=60
2	t=37
3	t=45
4	t=39
5	t=39
6	t=38

Figure 28

The application of the rule proceeds as follows:

<u>Step 1</u>. Construct an initial, though infeasible, schedule, by left-justifying all operations on all jobs as much as possible. Insert deadlines, and measure slack. (See figure 29, tableau 1.)

<u>Step 2</u>. Resolve conflicts on those machines having



82
20
18
16
14
12
10
TAB 1
AT COMPLETION OF J1M4
$Q_{M4} = 0$
TAB 2
AT COMPLETION OF J2M5
$Q_{M5} = 2$
TAB 3
19
8
JOB 6 HAS SMALLEST SLACK, SO SCHEDULE J6M5
TAB 4
AT COMPLETION OF J3M1
$Q_{M1} = 0$
TAB 5
AT COMPLETION OF J5M3
$Q_{M3} = 1$
SO SCHEDULE J2M3
TAB 6

FIG 29

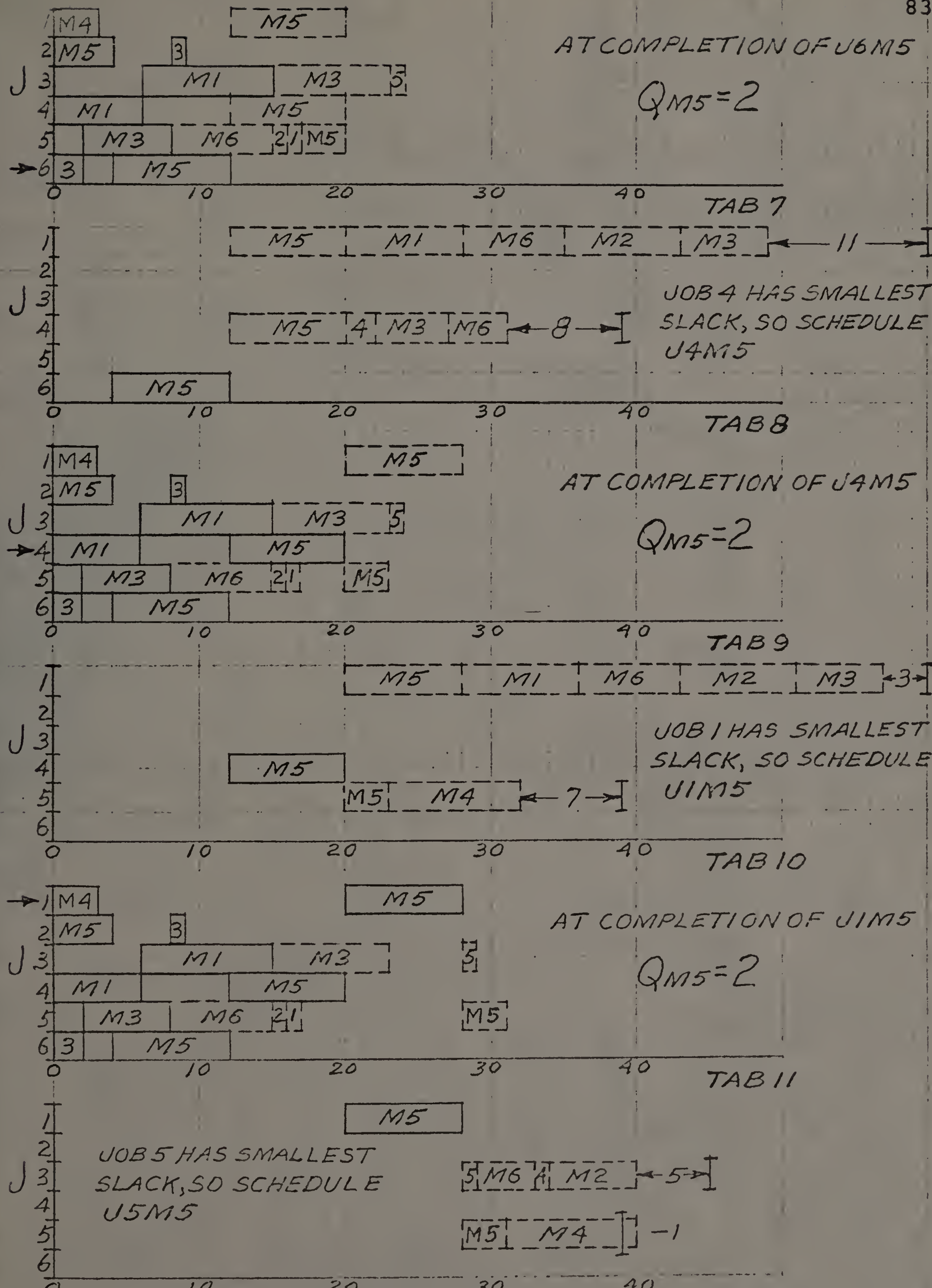
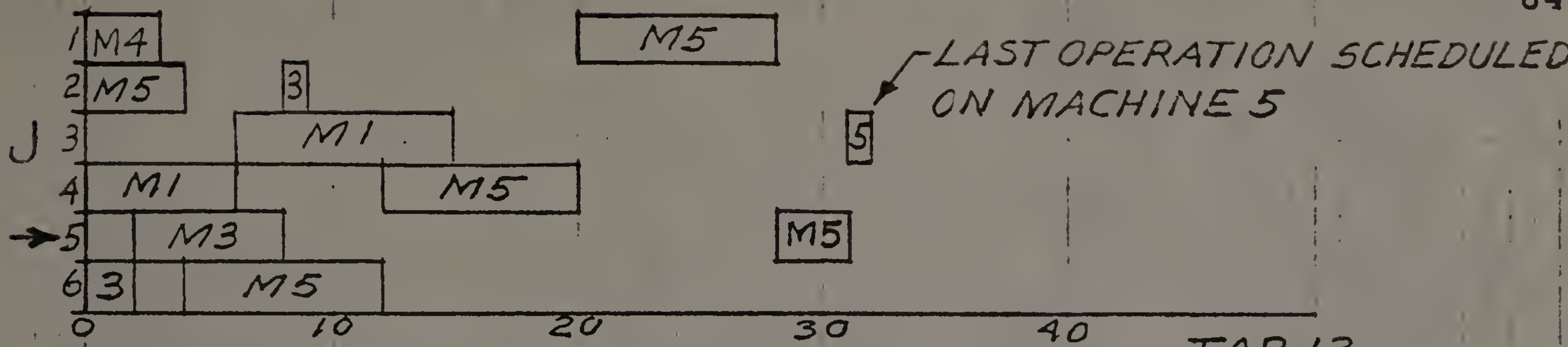
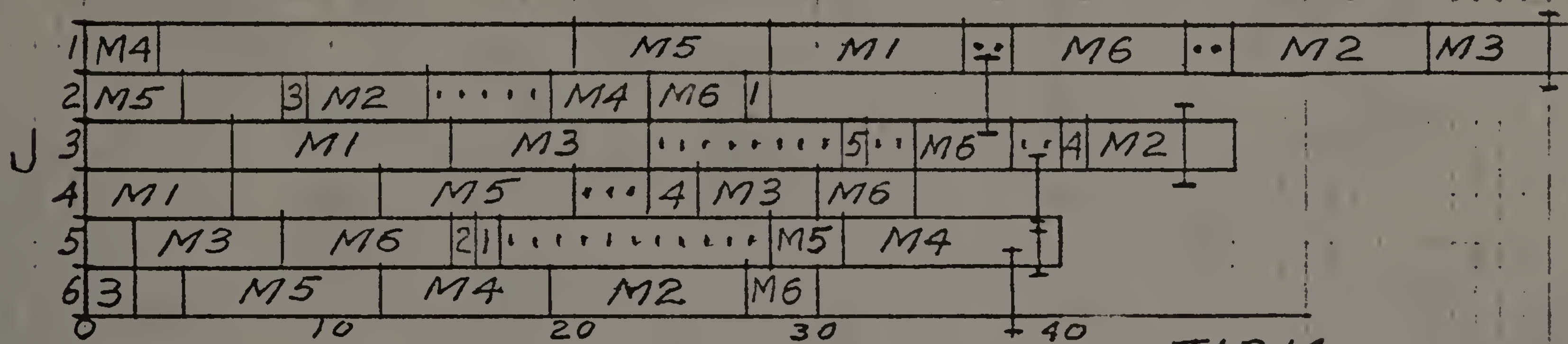

AT COMPLETION OF J6M5
QM5=2
TAB 7
JOB 4 HAS SMALLEST SLACK, SO SCHEDULE J4M5
TAB 8
AT COMPLETION OF J4M5
QM5=2
TAB 9
JOB 1 HAS SMALLEST SLACK, SO SCHEDULE J1M5
TAB 10
AT COMPLETION OF J1M5
QM5=2
TAB 11
JOB 5 HAS SMALLEST SLACK, SO SCHEDULE J5M5
TAB 12

FIG. 29 (CONT)



TAB 13



TAB 14

FIG. 29 (CONT)

queues of 2 or longer by giving first priority to the jobs with the smallest slack. At t=0 (figure 29, tableau 1), machines 1 and 3 each have queues of length 2. Of those jobs waiting to be processed on machine 1, job 4 has the smaller slack (14), so it is scheduled first. Of those jobs waiting to be processed on machine 3, job 6 has the smaller slack (10), so it is also scheduled first. (See figure 29, tableau 2).

Step 3. Having scheduled the first operation of each job, consider each of these operations to determine whether, at completion, a queue of 2 or more jobs will have formed at the facility. If conflicts occur, resolve them as in step 2.

To illustrate, at completion of J1M4 (figure 29, tableau 2), the queue length at machine 4 will be zero, since all M4 operations on other jobs must occur much later. Concern therefore passes to J2M5.

At completion of J2M5 (figure 29, tableau 3), the queue length at machine 5 will be 2. Specifically, the prior operations on jobs 1 and 6 have been completed prior to completion of J2M5; both jobs therefore await processing on machine 5.

In figure 29, tableau 4, the dispatching decision is made by examining slack for each of the competing jobs. Job 6 is found to have the smallest slack, so it is scheduled next on machine 5.

At completion of J5M3 (figure 29, tableau 6), the queue length at machine 3 will be 1. Since only job 2 awaits processing at this time, J2M3 is scheduled with no need to invoke the priority rule.

Sequential consideration of each of the job rows uncovers further conflicts that must be resolved. In particular, machine 5 exhibits a waiting line at the end of J6M5 (figure 29, tableau 7), J4M5 (tableau 9) and J1M5 (tableau 11). Dispatching decisions are made on the basis of tableaus 8, 10 and 12.

Step 4. Continue to examine job rows until it is determined that no further jobs await immediate processing on any facility (Q=0 for all jobs). Then, proceed to schedule the remaining operations around those already scheduled. If, in this process, new queuing problems arise, handle them in the manner previously described.

Figure 29, tableau 14 shows the completed schedule for the sample problem. Application of step 4 did not require the exercise of priority rules since in no instance did more than one job await processing at the end of any operation. (Q was always either 0 or 1.)

Tableau 14 indicates the relative successfulness of the job slack rule. While it was not possible to complete all jobs on time (jobs 1, 3 and 5 were late), the total lateness was quite small. The figure was 4 time units, as compared with 20 using the bottlenecks-first rule.

A logical criticism of the job slack rule is that it does not take into account the number of operations that remain to be processed beyond each decision point. There is assuredly a good deal of persuasion to the argument that large numbers of operations increase the likelihood of queue formation hence total waiting time for any given job. It therefore appears reasonable that number of remaining operations should be a factor in the determination of priorities.

Job Slack-Operation Quantity Rule

The job slack-operation quantity rule establishes priority on the basis of the ratio S/N, where S = slack and N = the number of operations remaining beyond the decision point. If N is held constant, jobs are afforded greater priority as S becomes smaller. On the other hand, if S is held constant, jobs receive higher priority as N increases. Therefore, in sum, the smaller the ratio of S to N, the higher the priority.

Application of the job slack-operation quantity rule proceeds in much the same fashion as that described for the job slack rule. The only difference is that conflicts are resolved by giving first priority to the jobs with the smallest slack per operation.

Resolution of the sample problem is shown in figure 30. In tableau 1, jobs 3 and 4 initially compete for machine 1. Since job 3 has the smallest slack per operation (2.66), it

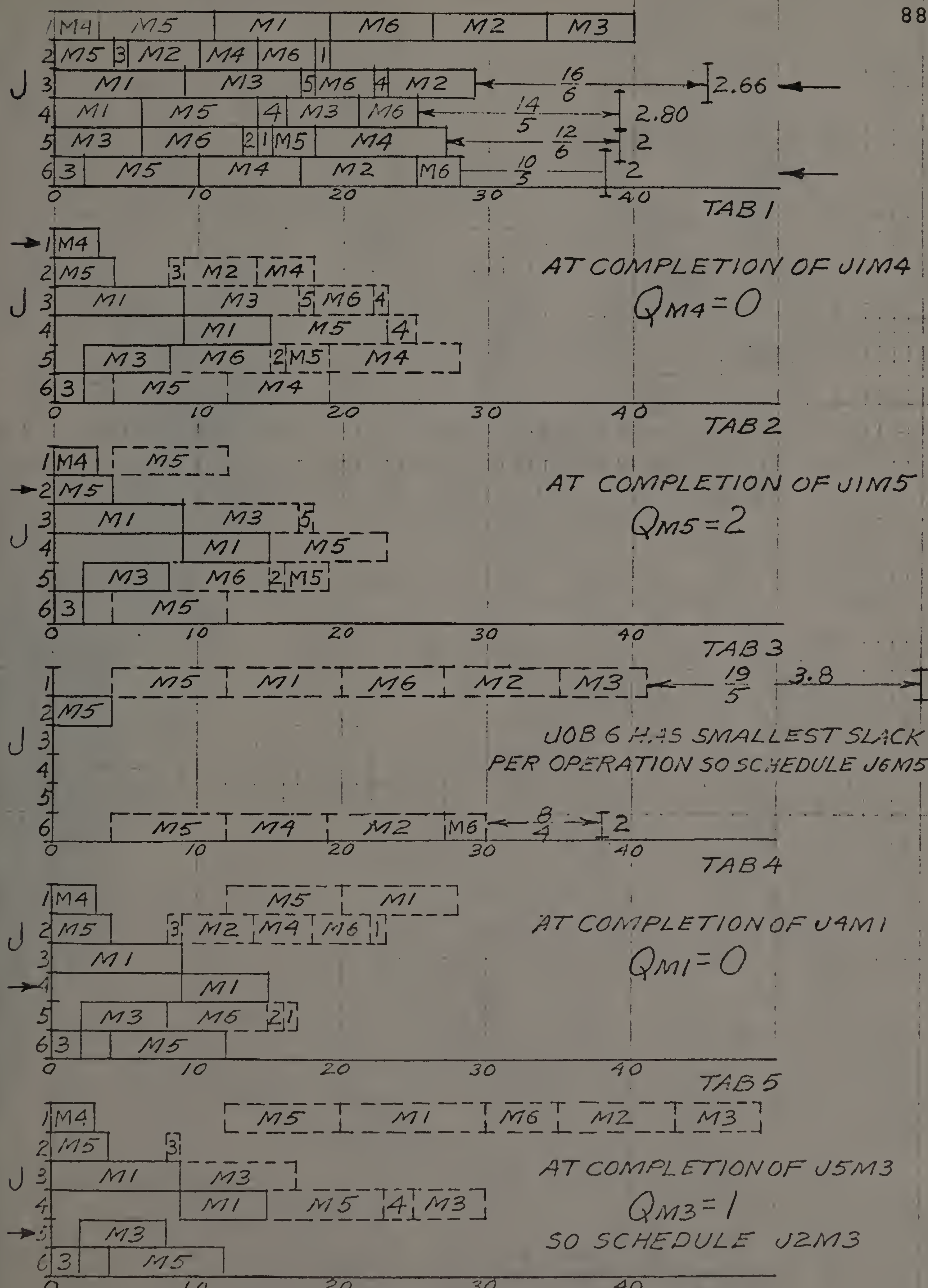

TAB 1
TAB 2
AT COMPLETION OF J1M4
QM4=0
TAB 3
AT COMPLETION OF J1M5
QM5=2
TAB 4
JOB 6 HAS SMALLEST SLACK
PER OPERATION SO SCHEDULE J6M5
TAB 5
AT COMPLETION OF J4M1
QM1=0
TAB 6
AT COMPLETION OF J5M3
QM3=1
SO SCHEDULE J2M3

FIG. 30

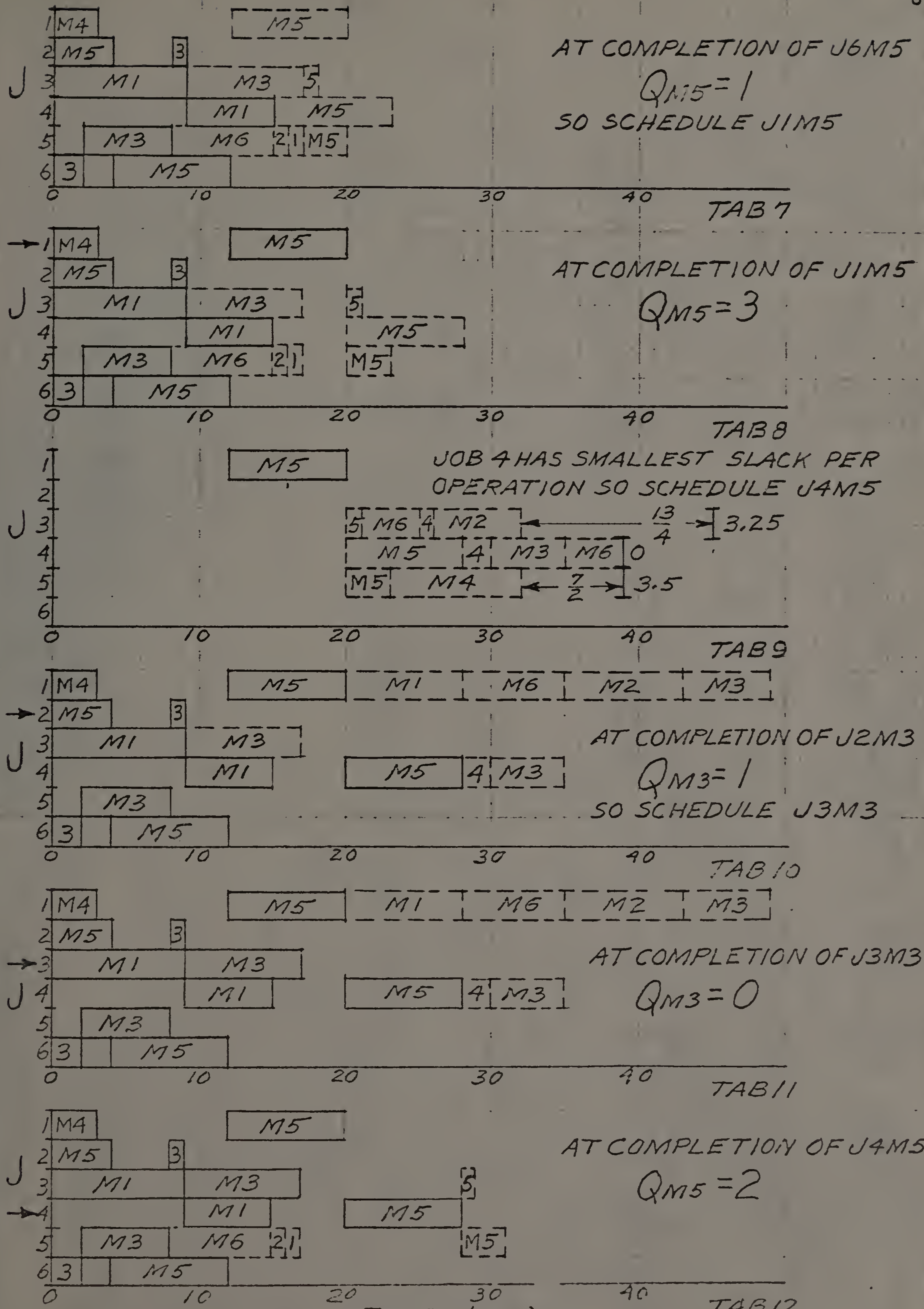

AT COMPLETION OF J6M5
$Q_{M5} = 1$
SO SCHEDULE J1M5
TAB 7
AT COMPLETION OF J1M5
$Q_{M5} = 3$
TAB 8
JOB 4 HAS SMALLEST SLACK PER OPERATION SO SCHEDULE J4M5
13/4 3.25
7/2 3.5
TAB 9
AT COMPLETION OF J2M3
$Q_{M3} = 1$
SO SCHEDULE J3M3
TAB 10
AT COMPLETION OF J3M3
$Q_{M3} = 0$
TAB 11
AT COMPLETION OF J4M5
$Q_{M5} = 2$
TAB 12

FIG. 30 (CONT)

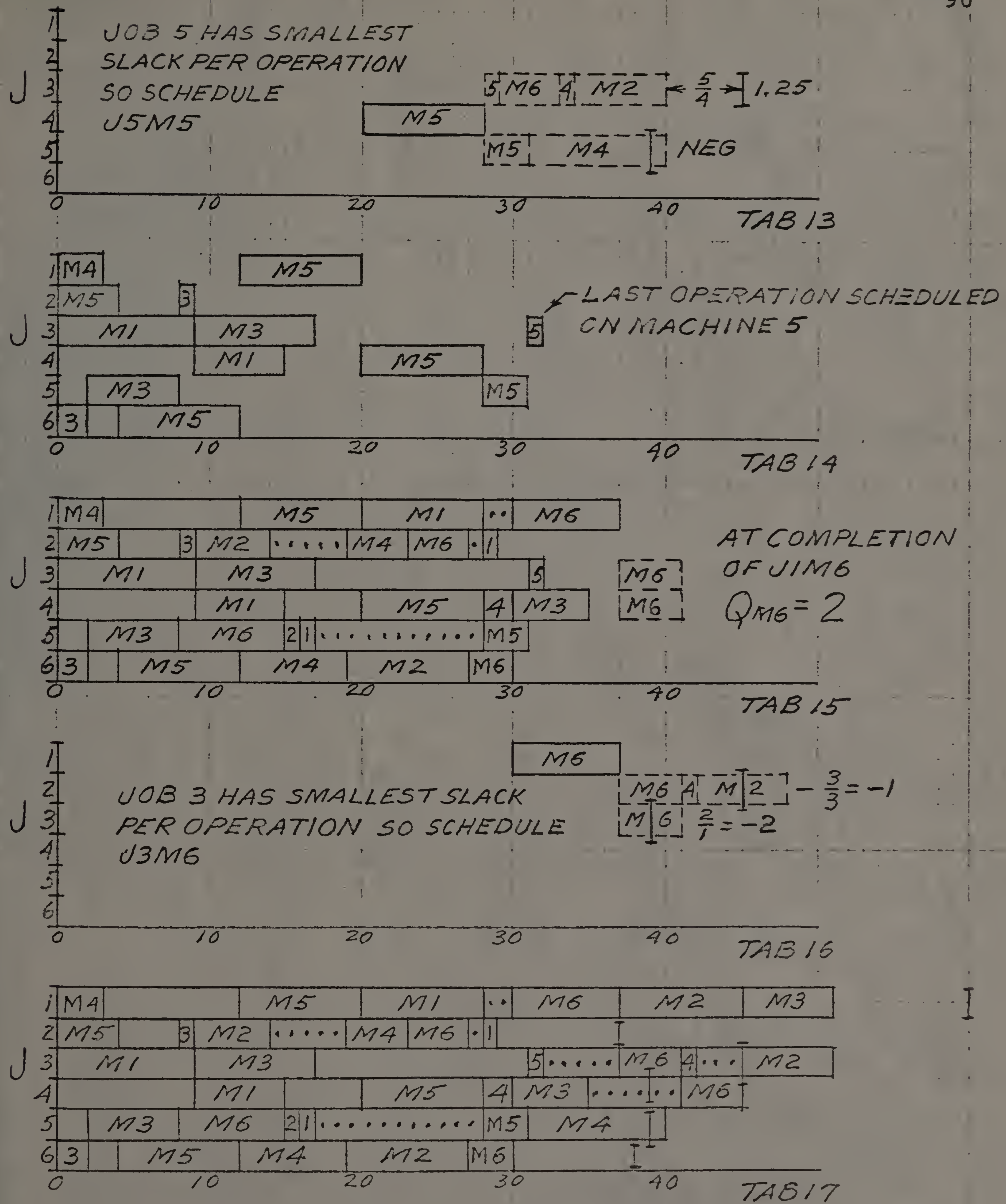

JOB 5 HAS SMALLEST SLACK PER OPERATION SO SCHEDULE J5M5
5/4 1.25
NEG
TAB 13
LAST OPERATION SCHEDULED ON MACHINE 5
TAB 14
AT COMPLETION OF J1M6
QM6 = 2
TAB 15
JOB 3 HAS SMALLEST SLACK PER OPERATION SO SCHEDULE J3M6
- 3/3 = -1
2/1 = -2
TAB 16
TAB 17

FIG. 30(CONT)

is processed first. Jobs 5 and 6 initially compete for machine 3, but slack per operation is the same (2) for both. Discrimination is therefore arbitrarily made on the basis of slack alone, and job 6, having a slack of 10, receives first priority.

Tableaus 3, 8 and 12 show instances in which queues of two or more jobs have formed in front of machines. In each of these instances, dispatching is accomplished via the job slack-operation quantity rule. (See tableaus 4, 9 and 13.)

Tableau 15 illustrates application of the latter part of step 4 (see page 86). A queuing problem arises at the end of J1M6, but resolution is easily accomplished (tableau 16). The ultimate schedule is shown in tableau 17.

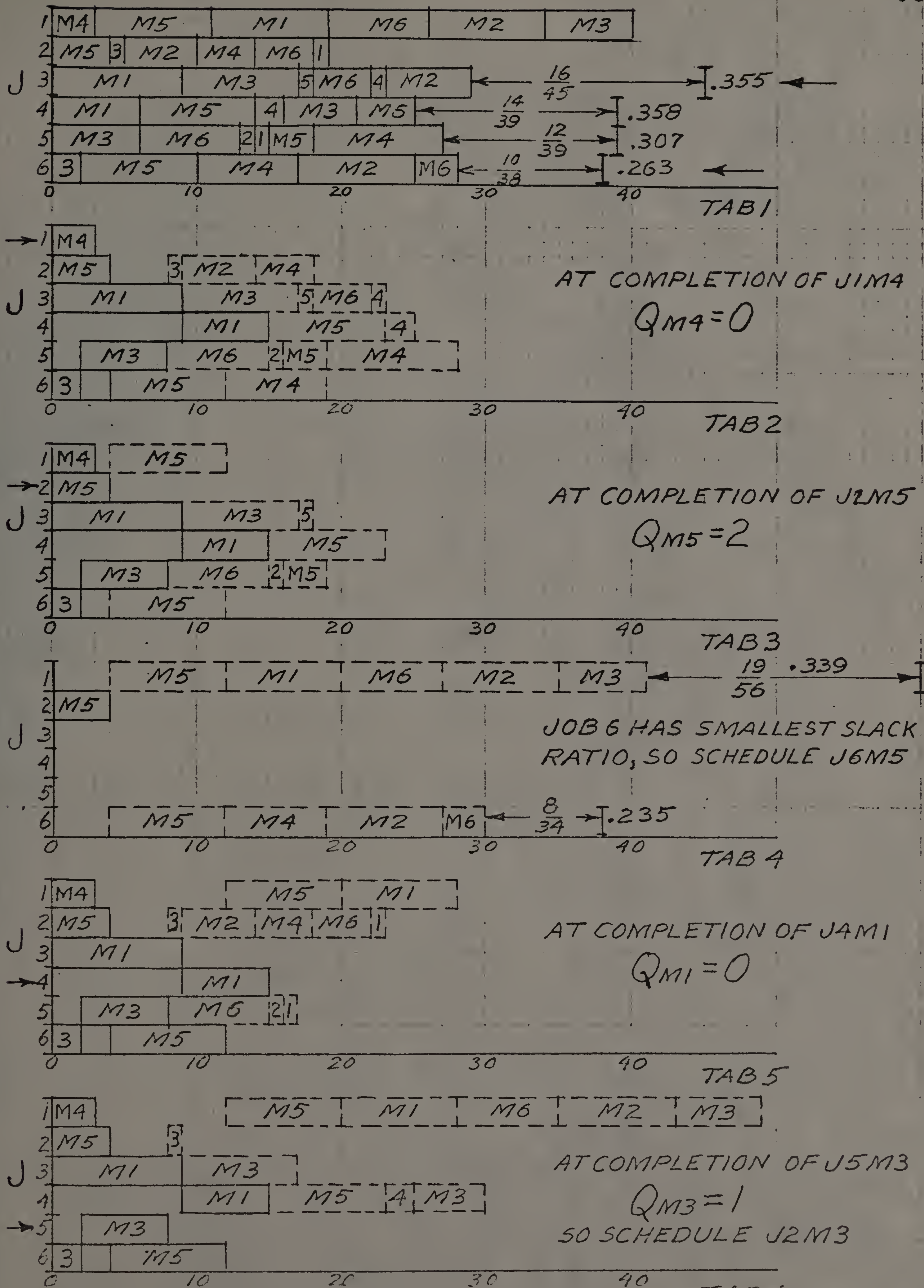
For the problem under consideration, the job slack-operation quantity rule did not perform as well as the less-sophisticated job slack rule. Again, three jobs came in late, but the total lateness was 13, compared to only 4 for the job slack rule. While generalization would evidently be preposterous, this example clearly illustrates the basic nature of priority rules. A rule that may produce exemplary results in one instance may not work nearly as well in another.

Job Slack-Total Remaining Time Rule

This rule asserts that the priority of jobs ought to be judged on the basis of slack in relation to total remaining pre-deadline time. Specifically, the measure is S/T, so that what emerges is merely slack as a percentage rather than an absolute value. The smaller this percentage, the higher the priority.

Resolution of the sample problem, using the job slack-total remaining time rule, is shown in figure 31. In tableau 1, percentages are taken for each of the conflicting jobs. Job 3, for example, has 45 remaining minutes, measured from t=0, in which to be accomplished. Of this total time, 16 minutes is slack. Therefore, slack is 16/45 or 35.5% of total remaining pre-deadline time. Similar computations, performed for the other conflicting jobs, lead to the conclusion that jobs 3 and 6 should receive first priority on machines 1 and 3, respectively.

The remaining resolution stages are substantially identical to those of figure 30. Applications of the job slack-total remaining time rule (tableaus 4, 9, 13) result in dispatching decisions that are identical to those resulting from application of the job slack-operation quantity rule. Also, the ultimate schedules are precisely the same.



TAB 1
16/45 .355
14/39 .358
12/39 .307
10/38 .263
TAB 2
AT COMPLETION OF J1M4
$Q_{M4}=0$
TAB 3
AT COMPLETION OF J2M5
$Q_{M5}=2$
TAB 4
19/56 .339
JOB 6 HAS SMALLEST SLACK RATIO, SO SCHEDULE J6M5
8/34 .235
TAB 5
AT COMPLETION OF J4M1
$Q_{M1}=0$
TAB 6
AT COMPLETION OF J5M3
$Q_{M3}=1$
SO SCHEDULE J2M3

FIG. 31

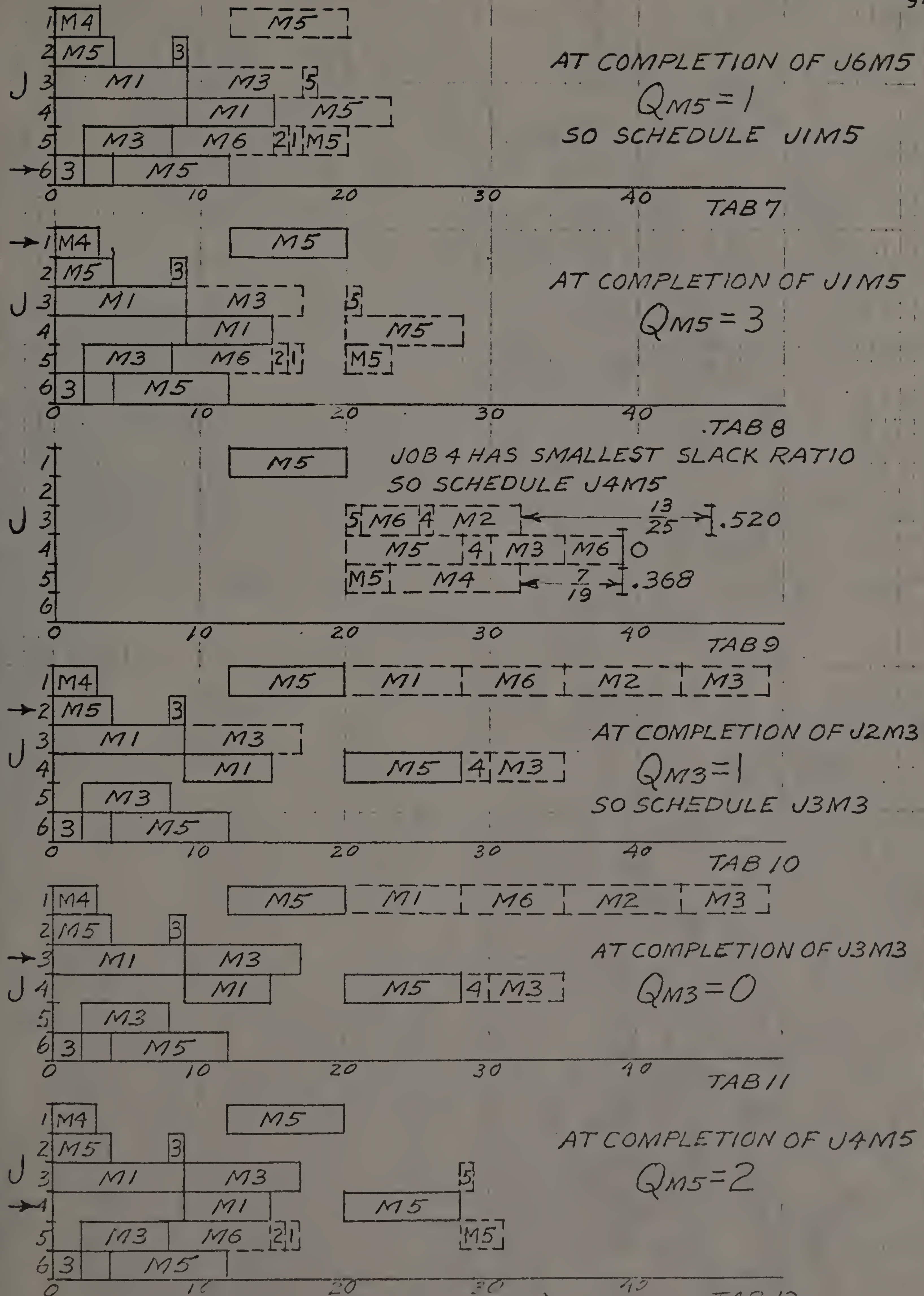

AT COMPLETION OF J6M5
QM5 = 1
SO SCHEDULE J1M5
TAB 7
AT COMPLETION OF J1M5
QM5 = 3
TAB 8
JOB 4 HAS SMALLEST SLACK RATIO
SO SCHEDULE J4M5
13/25
.520
0
7/19
.368
TAB 9
AT COMPLETION OF J2M3
QM3 = 1
SO SCHEDULE J3M3
TAB 10
AT COMPLETION OF J3M3
QM3 = 0
TAB 11
AT COMPLETION OF J4M5
QM5 = 2
TAB 12

FIG. 31 (CONT)

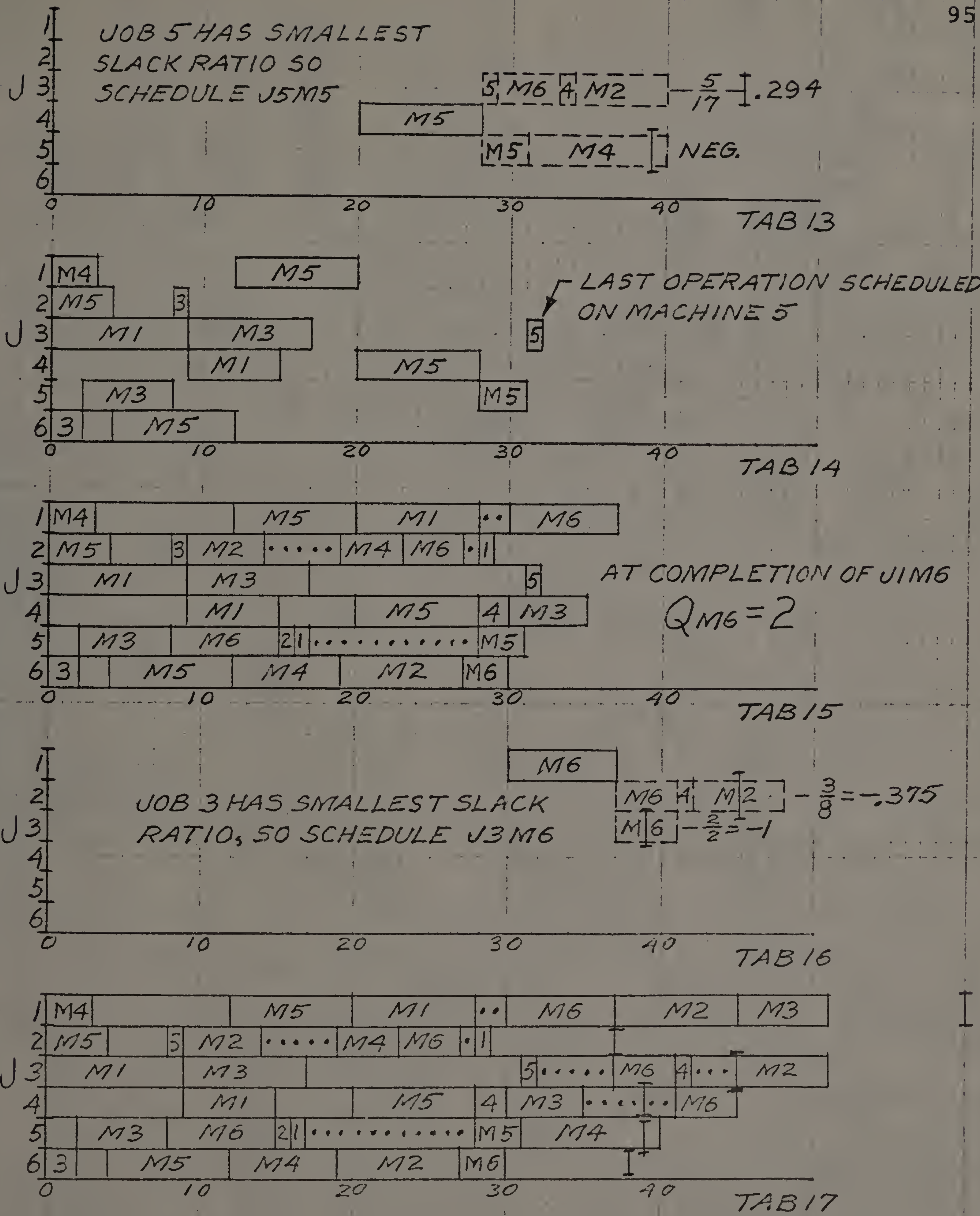

JOB 5 HAS SMALLEST SLACK RATIO SO SCHEDULE J5M5
.294
NEG.
TAB 13
LAST OPERATION SCHEDULED ON MACHINE 5
TAB 14
AT COMPLETION OF J1M6
QM6 = 2
TAB 15
JOB 3 HAS SMALLEST SLACK RATIO, SO SCHEDULE J3M6
-.375
-1
TAB 16
TAB 17

FIG. 31 (CONT)

CHAPTER V

THE APPLICATION OF SUPPLEMENTAL HEURISTICS

Introduction

Thus far, we have considered three priority rules whose purpose was to cause the completion of jobs on or before their due dates. Evidently, we have not exhausted all possibilities; indeed, an endless number of sophisticated rules might be concocted, each supported by no small amount of compelling logic.

On an intuitive basis, however, one can sense the futility of any venture bent on discovery of the "perfect" decision rule. Any rule, whether simple of compound, simple-minded or sophisticated, cannot, by nature, be the best rule in all cases. This is because priority rules are used, by admission, as imperfect substitutes for a yet unknown precise methodology. Given that all priority rules are of an imperfect gendre, the quest for a perfect one seems pointless.

The general effectiveness of priority rules is limited by their localized nature, that is, they permit decisions to be made at any given problem stage on the basis of local information only. No account is taken of future obstacles that may arise directly from the decision itself. In some instances, therefore, blind application of a certain priority rule will ultimately be revealed as having done more harm than good.

In light of the foregoing, a reasonable course of action would seem to be to use the simplest priority rules as guidelines for scheduling, but to temper these rules according to the parameters of each individual problem. Such a procedure would recognize "customization" as being superior to the use of very complicated decision rules.

Clearly, not all problems would require the same degrees of divergence from "blind" priority rule application. There are, however, comparatively few instances in which the scheduling process would not be expected to benefit, at least marginally, from the judicious application of certain supplementary heuristics.

The extent to which the schedule is able to efficiently "customize" will depend largely on his skill at spotting trouble before it occurs. This ability varies from scheduler to scheduler, but tends to decline as the size of the schedule increases. Even the most adroit trouble-shooters are likely to miss certain opportunities for improvement as schedules grow very large.

We shall now examine several customization techniques in the context of the sample problem. It will be recalled that no previously tested priority rule was successful, by itself, in generating a schedule that would permit all jobs to be completed on time. In each instance, several jobs came in late, although the total lateness was appreciably less when the job slack rule was used.

Using the job slack rule as a primary rule, we now attempt to "rebuild" the final schedule such that all jobs are completed on or before their due-dates. Several supplemental rules are considered.

Alternative Assignment Rule

If dispatching according to the primary rule necessarily causes any job, whether in queue or not, to become late, consider the reassignment of priorities.

A specific application procedure for the alternative assignment rule is as follows:

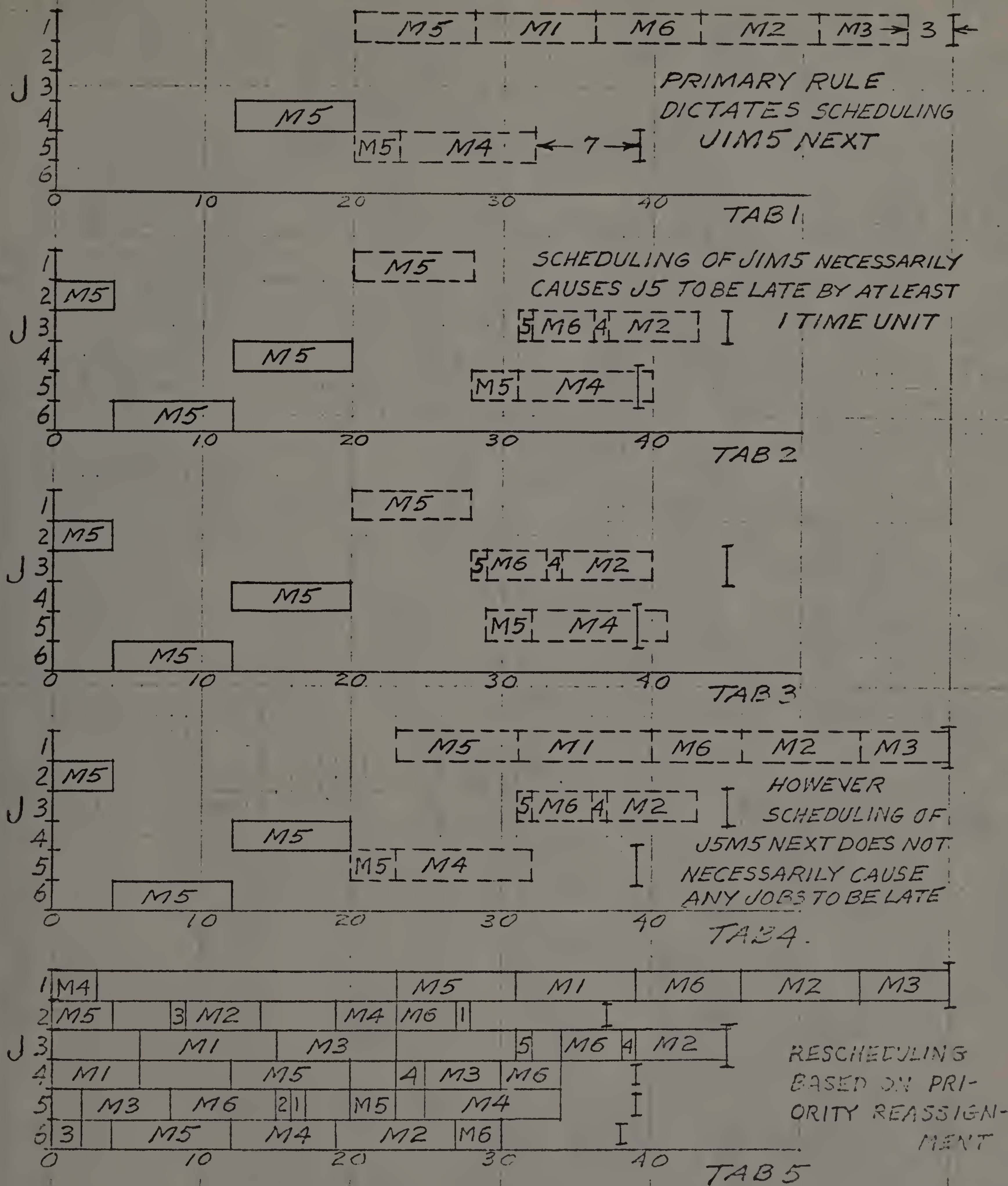
(1) Tentatively dispatch job j to facility f by invoking job slack or some other primary dispatching rule.

(2) Check to see if such decision, <u>in and of itself</u>, occasions the lateness of any job.

(3) If a lateness occurs, temporarily revoke the original decision and examine the consequences of alternative assignments to the facility in question.

(4) If an alternative assignment produces superior results (i.e. no latenesses or smaller latenesses), abandon the original rule and maintain the alternative assignment. Otherwise, restore the original assignment based on the original rule.

<u>Jobs in queue</u>. To illustrate application of the alternative assignment rule, consider figure 32, in which tableau 1 is a repeat of figure 29, tableau 10. In the original an-



PRIMARY RULE DICTATES SCHEDULING J1M5 NEXT
TAB 1
SCHEDULING OF J1M5 NECESSARILY CAUSES J5 TO BE LATE BY AT LEAST 1 TIME UNIT
TAB 2
TAB 3
HOWEVER SCHEDULING OF J5M5 NEXT DOES NOT NECESSARILY CAUSE ANY JOBS TO BE LATE
TAB 4
RESCHEDULING BASED ON PRIORITY REASSIGNMENT
TAB 5

FIG. 32

alysis, the job slack rule was blindly applied with no attempt being made to determine the long-run consequences of such application. On this basis, a decision was made to schedule J1M5.

Tableaus 2 and 3 of figure 32 examine the ultimate consequences of the decision to schedule J1M5. If J1M5 is scheduled as shown, job 5 must necessarily be late by either 1 or 2 time units, depending on subsequent sequencing of J3M5 and J5M5. This finding causes the immediate invocation of the alternate assignment rule.

In tableau 4, it is determined that an alternative to the original priority assignment produces a superior schedule. Specifically, if J5M5 is scheduled in place of J1M5, with J1M5 and J3M5 directly following in that order, no latenesses are generated. The primary rule is thus subordinated to the secondary rule, and the alternative configuration is accepted.

Tableau 5 shows figure 29, tableau 14, modified to show the reassignment of priorities. All jobs are now completed on or before their due-dates.

Some weaknesses in our methodology should be noted. First, the process of "looking ahead" is a highly selective one: we examine some things, and deliberately ignore others. Clearly, this approach is based on the recognition that we cannot possibly hope to anticipate all of the ramifications of a given dispatching decision. To do so would require enumeration of all feasible schedules, a task which has been dismissed as impossible.

To cite one example of this "selectivity" we note that, in the problem at hand, no attempt has been made to assess increases in lateness that result from elimination of conflicts on machines not presently under consideration. Tableaus 2 and 3, for example, reveal that attention has been confined to latenesses occasioned by certain dispatching decisions involving machine 5 only, even though it is fully recognized that subsequent elimination of conflicts on other machines (e.g. M4) will create additional latenesses. (The assumption is, of course, that M4 may eventually be treated in isolation, in the same manner that M5 was, but there is evidently a logical weakness to such "piecemeal" analysis.)

A second major difficulty involves the ability of the scheduler to apply the alternative assignment rule in problems of very large scale. Once the primary rule has been invoked, the scheduler must then test for latenesses of all jobs remaining to be scheduled on the machine in question. If there are even 4 such jobs, the scheduler would legitimately like to examine as many as 4! = 24 different configurations in order to determine the applicability of secondary rule invocation. Furthermore, if the 24th trial fails to generate a schedule with no latenesses, the scheduler must then test each of the 4 alternative dispatching decisions. In each case, the scheduler may revert to the primary rule (e.g. job slack), thus avoiding the need to evaluate (in this case) 24 different configurations for each of the 4 alternative dispatching decisions. (Reversion to the primary rule is frequently a satisfactory substitute for enumeration. See, for example, the schedule of figure 32, tableau 4, in which the primary rule was invoked directly.)

If successive reversions to the primary rule seem unappealing, an alternative approach is to informally "sample" from among all possible schedules. Suppose, for example, that the scheduler is at stage 2 of the application procedure for the alternative assignment rule. Suppose, furthermore, that 4 jobs remain to be scheduled on the machine in question, thus suggesting the need to enumerate 4! schedules of the type in tableaus 2 and 3, figure 32, in order to determine the applicability of the secondary rule. The scheduler might circumvent this apparent difficulty by trying at random, say, 4 or 5 of the 4! possible schedules. If, within this sample, he encounters a lateness-free schedule, he need go no further. If, on the other hand, each selection produces a lateness, he might infer that the population contains no lateness-free schedules and proceed to invoke the secondary rule.

If his inference is incorrect, that is, if the secondary rule did not need to be invoked when indeed it was, the penalties are not usually severe. There is still a good chance of coming up with a "good" solution, even though a better one might have been passed up as a direct consequence of the sampling.

Jobs not in queue. We have dealt with an example involving the reassignment of priorities for jobs currently in queue at a given machine. On occasion, we will wish to consider another possibility, namely, that some job, not currently in queue at that machine, should, nevertheless, deserve first priority.

To illustrate this possibility, consider the schedule of figure 33 with due-dates as shown. (This schedule is unrelated to the previous sample problem.) At the present time t_p, three jobs are in queue at M_C, namely, J_C, J_D and J_E. If job slack is invoked as the primary rule, J_CM_C is scheduled first.

The scheduling of J_CM_C immediately precludes the completion of job B on or before its deadline. Specifically, the scheduling of J_CM_C causes job B to be completed, at minimum, 4 days beyond its due date.

Although JB is not in queue at machine C, there is every reason, on the basis of a localized analysis, to schedule this job in place of J_C, even though machine C must be kept idle between t_p and t_f. (J_BM_C is "locked in" by previously scheduled operations, and cannot begin prior to t_f.) If we find that no conflicts occur among operations subsequent to J_CM_C, J_CM_C and J_EM_C, and that further "lengthening" of the schedule is therefore not required, it is

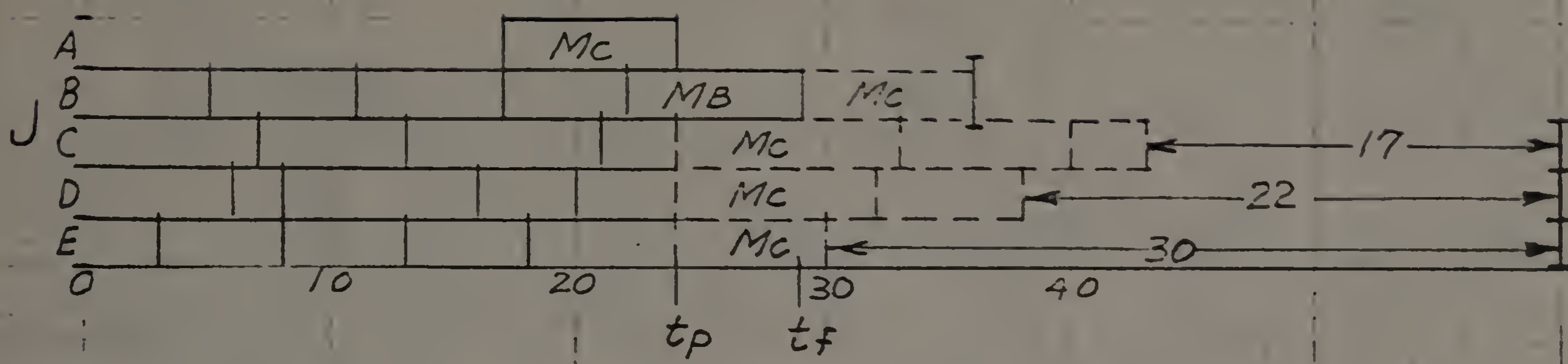


FIG. 33

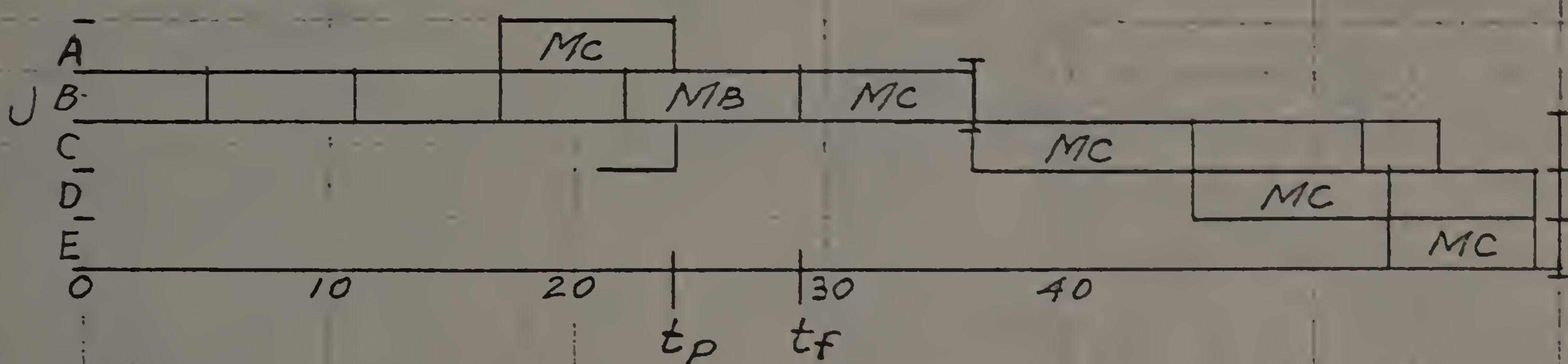


FIG. 34

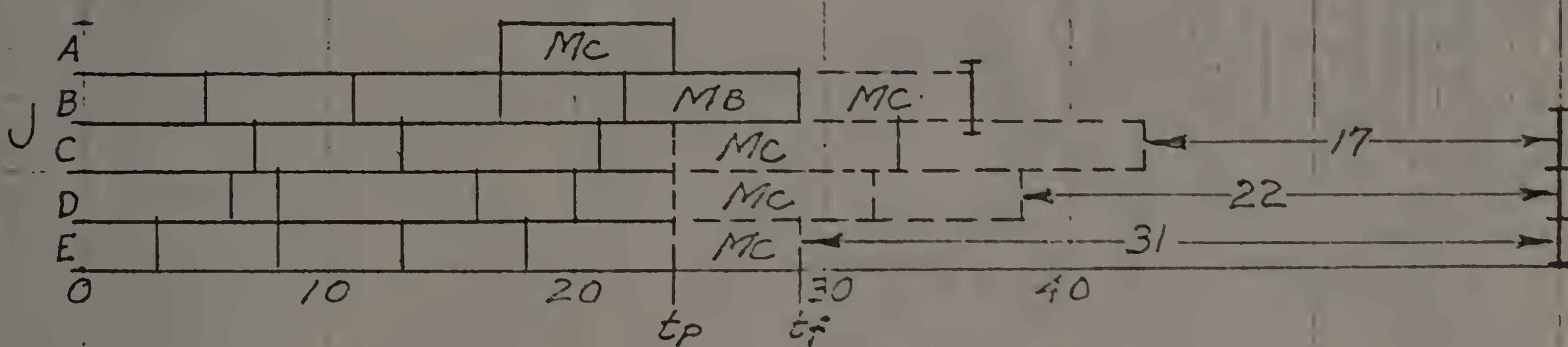


FIG. 35

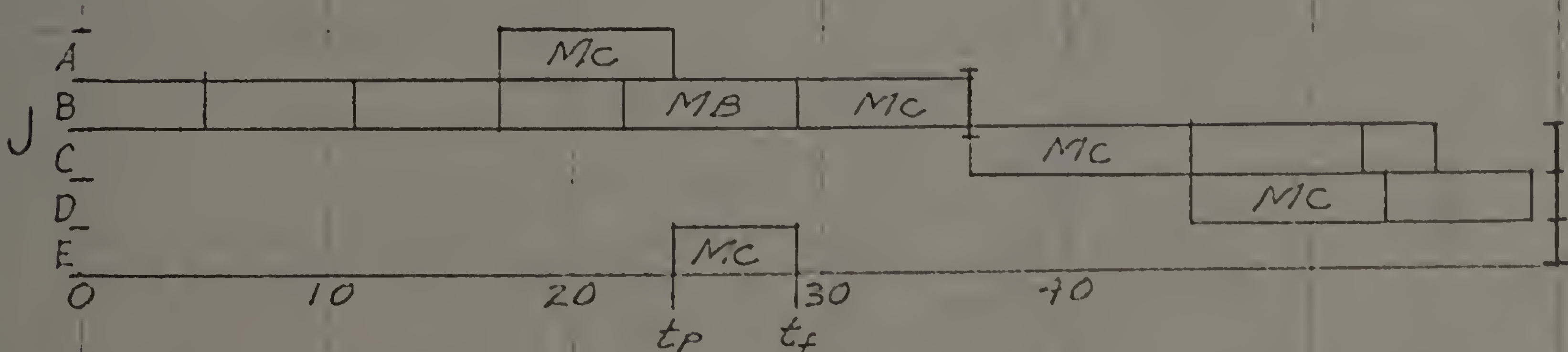


FIG. 36

clear that we have not had to "rob Peter to pay Paul," that is, giving first priority to J_BM_C has not caused any other job to come in late. (A schedule which exhibits no latenesses at all is shown in figure 34.) On the other hand, if looking ahead a short distance reveals serious conflicts, a reversion to the original rule may be in order.

Idle Time Reduction Rule

In the example at hand, the immediate scheduling of job B on machine C creates an idle time "gap" $(t_f - t_p)$ that cannot be filled by any other waiting job. (Operations J_CM_C, J_DM_C and J_EM_C are all too long.) However, this unfortunate situation is not necessarily typical. Suppose, for example, that J_EM_C were of slightly different dimensions (see figure 35). Now J_EM_C could be "inserted" prior to commencement of J_BM_C, with a corresponding reduction in idle time, as well as a much earlier completion time for job E. (See figure 36.)

The idle time reduction rule might be formalized as follows: Wherever idle time "gaps" are found to exist in any schedule, try to fill these "gaps" with the longest operations available, but do not reschedule unless such modification permits operations to begin sooner rather than later.

Several important implications arise from the basic statement of the rule:

(1) If several different operations are in contention for filling a particular gap, the one that fills the gap most

completely should be chosen.

(2) Rescheduling of operations for the purpose of reducing idle time should only be undertaken in the event that the rescheduling permits the operations to begin sooner rather than later. If the reverse is true, the net effect will be to delay completion of the rescheduled job.

(3) Care must be taken not to disturb the ordering of operations. In general, the decision must be examined within the context of its relationship to every other immediate and future (as far as is practicable) aspect of the problem.

It is extremely important to note that application of the idle time reduction heuristic generally supposes prior application of the alternative assignment heuristic. A fundamental aspect of the alternative assignment heuristic is that it requires the scheduler to peer into the future, at least a short distance, in order to assess the consequences of giving first priority to some job not currently in queue. If these consequences are held to be desirable, the job is scheduled, and an idle time gap immediately appears. (This gap would not have been generated if the job had been selected from queue.) Clearly, it is then desirable to invoke the idle time reduction rule in the hopes of filling or partially filling the vacancy.

The interrelationship between the alternative assignment and idle time reduction heuristics can easily be understood if there is some prior understanding of the difference between basic priority rules (such as "shortest imminent operation") and the alternative assignment rule. Common to all basic priority rules is a modus operandi which requires the scheduler to make decisions on the basis of immediate local information only. At any decision point, the only possible choices for scheduling are those jobs presently in queue; no attempt is made to investigate the desirability of scheduling jobs outside this set. Furthermore, in the course of applying the primary rule, the scheduler never looks ahead. He blindly continues to apply the rule until all jobs have been fully scheduled.

In contrast to all basic priority rules, the alternative assignment rule has a "look-ahead" feature. Rather than confining interest only to jobs in queue, the scheduler investigates beyond the immediate "area" in the hopes of detecting troubles (latenesses, serious conflicts, etc.) before they occur.

If concern is for the future rather than the present, there naturally tends to be an oversight of things in between. Having looked to the future, it is thus necessary for the scheduler to go back and examine the interim. It is in the course of this examination that the idle time reduction heuristic will be utilized.

In summary, then, the alternative assignment rule has certain features which dictate subsequent invocation of the idle time reduction heuristic. These features are not shared by any of the basic priority rules.

The alternative assignment and idle time reduction heuristics together constitute a very powerful device for improving schedules. Under certain conditions, however, it may be desirable to consider other possibilities.

The imposing of artificial constraints or the temporary relaxing of real ones are two devices which are commonly used to "force" schedules into conformity with requirements. Rules illustrating each of these approaches are discussed next.

Time Constraint Relaxation Rule

In the original statement of the general job-shop scheduling problem, it was assumed that all processing times were of fixed length, and that these exact time requirements would have to be rigidly acknowledged in the scheduling of jobs. Having made this assumption, the "compression" of operation lengths was held to be impossible; if an operation was too long to fit a given "gap," it would simply have to be delayed until an adequate opening became available.

In practice, the fixed-processing-time restriction is frequently violated. Under normal conditions (i.e. in the absence of improperly running machines or outright breakdowns), some variability of processing times would be expected to be encountered, largely due to performance variances among machine operators. The "fixed" processing times of any example are assumed to be based on averages or standards; depending on the rate of output of the individual workers, actual processing times might be expected to fall above or below the norm.

It is not our intention to engage in a discussion regarding the likelihood of encountering favorable or unfavorable variances, but merely to point out that it is generally possible to influence the extent of such variances through the exertion of managerial control. To be specific, it is probably quite possible to decrease processing times slightly, especially if the need for improvement is only temporary. Temporary improvements in productive efficiency can be effected in many ways; however, if economies are to be confined to selected areas for limited periods of time, the best approach is probably to juggle personnel assignments in such a way that the fastest workers are always assigned to those jobs (or operations) which management deems "critical."

If we assume that it is at least occasionally possible to complete operations in less than the specified times, we are led to the conclusion that it is also possible, on occasion, to "squeeze" an operation into an idle time gap that is slightly too short. Under these circumstances, the efficiency of a given schedule is subject to radical improvement.

The time-constraint-relaxation rule may be formally stated as follows: If an operation might profitably be inserted between two others, but if its length is slightly too long, reduce the length appropriately, and insert the operation.

Two important questions logically develop from this statement: (1) What is implied by the adjective "slightly"? and (2) Is there a practical limit on the number of times the rule can be applied in any one problem?

In answer to the first question, there are evidently restrictions on the extent to which a given operation may be presumed "flexible." It would be difficult, for example, to envision a 60 minute operation being completed in 5 minutes, even under the best possible circumstances. In this instance, it is rather clear that such gross "compression" should not be attempted; yet there are many other cases in which the answer is not so clear cut. In these cases, some criterion must be developed to permit the making of decisions.

It would be highly presumptious to assume that such a criterion could be established without some intimate knowledge of each specific work environment. Depending on worker attitudes and abilities (and, of course, to a degree on machine efficiency), one would expect minimum processing times to vary widely. In some cases, a 20 or 30% reduction over standard might not be at all unreasonable; in other cases, it would clearly be excessive.

The meaning of the word "slightly," therefore, can only be determined in a specific context. Successful application of the time constraint relaxation rule must therefore engender a thorough study of the work situation, in order to determine maximum possible percentage reduction in procsssing times.

In answer to the second question, it is clear that the more often the time constraint relaxation rule is applied in any given problem, the greater the demands placed on management's ability to effect reductions in processing times. Reductions on one or two operations may be easy to achieve in the manner previously described; on the other hand, any shop has a limited number of "best" personnel. If processing times on a great many operations must be simultaneously reduced, the allocation problem is compounded immensely. It may turn out that there is simply not enough "talent" to go around.

Again, the permissibility of repeated applications of the time constraint relaxation rule can only be determined after intensive study of the specific situation in which the rule is to be employed. In some shops, only one or two applications may be allowable; in others, it may be possible to slightly reduce processing times of many operations without placing unacceptable demands on personnel and/or machines.

The utilization of the time-constraint-relaxation rule can easily be illustrated with reference to the most recently discussed example. In figure 33, looking ahead suggests the advisability of scheduling J_BM_C next. However, if J_BM_C

is so scheduled, a vacancy of length $(t_p - t_f)$ develops which, as the problem stands, cannot be filled.

In figures 35 and 36, we assumed a different problem, that is, we assumed M_C was given as 5 time units in length rather than 6. This new assumption made it possible to "insert" $J_E M_C$ prior to $J_B M_C$, thus illustrating the idle time reduction heuristic.

Suppose now, however, that we assume a "compressability" feature, that is, we now assume that it is possible (through reassigning of personnel, special controls, etc.) to process an operation that would normally take 6 time units in only 5. This would amount to a temporary processing-time reduction requirement of only about 17%, a figure which, on the average, would not be deemed excessive. Thus, the operation would be scheduled <u>as if it consumed</u> only 5 minutes instead of 6, in the manner suggested in figure 36.

Occasionally, the actual performance of operations may reveal that the "compressability" assumptions were not warranted. (Plans may have gone awry; perhaps it was not possible to achieve the desired allocation of workers.) In any event, the consequences will probably not be disastrous, especially if the failure is not on any grand scale. (If the compressability assumption is found to be unjustifiable in only one or two cases out of many, and if the assumption was not an extravagant one to begin with, difficulties will generally be slight.)

To illustrate, suppose the 17% reduction of the immediate example turned out to be a figment. Under those circumstances, the start of $J_B M_C$ would simply have to be advanced by 1 time unit, thus engendering 1 unit's lateness in completion of the job. Furthermore, all subsequent operations on other jobs would have to be advanced by as much as one day, in order to accommodate the unforeseen lengthening of $J_E M_C$.

In this example, it is clear that such modifications are only slightly disruptive. Job B is now completed 1 day past its due date, but all other jobs are still finished on time. Additionally, job E is still completed some 30 days early, conceivably a matter of some positive importance. Comparison between figures 34 and 35 indicates that this earliness would not have occurred without the assumption of compressability.

In closing, it should be noted that the time constraint relaxation heuristic is directly supplementary to the idle time reduction heuristic. In instances where stated processing times appear to prohibit "insertion" of operations, invocation of the time constraint relaxation heuristic follows as a logical consequence.

Artificial Deadline Rule

If repeated application of the previously-discussed heuristics fails to bring about a satisfactory schedule in terms of the meeting of due dates, it is frequently useful to impose artificial due-dates of a more restrictive nature, and then reschedule. The imposition of these pseudo-deadlines, which are used as a basis for recomputing slack, has the effect of contracting the schedule such that the likelihood of meeting real deadlines is increased.

A formal statement of the steps involved in application of the rule is as follows:

(1) If the original schedule fails to complete all jobs on or before their due dates, establish a pseudo-deadline for each offending job at a point in time equal to the real (original) deadline, minus the amount of the lateness. (For example, if a job has an original deadline at T, and if the job is originally completed t days late, establish the pseudo-deadline at T-t.)

(2) Reschedule all jobs, using the pseudo-deadlines as a basis for application of the job slack or similar rule.

(3) Evaluate the new schedule on the basis of the original deadlines.

(4) Repeat the procedure as desired.

The artificial deadline procedure has the effect of forcing the latest jobs into positions of highest priority each time the scheduling process is repeated. To illustrate this and other points, we return to the example that was developed extensively in chapter four. It will be recalled that application of the job slack rule without invocation of supplemental heuristics originally led to the schedule of figure 37, tableau 1 (figure 29, tableau 14 repeated). In this schedule, jobs 1, 3 and 5 are all completed past their due dates.

In tableau 2 of figure 37, the first set of pseudo-deadlines is imposed, according to step 1 of the artificial deadline rule. Job 1 was originally one day late, so its due date is moved back by one day. Similarly, the due-dates of jobs 3 and 5 are moved back by two and one days respectively.

Tableaus 3, 4, 5 and 6 depict the critical stages in the development of the potentially improved schedule (tableau 7). It is clear, however, that the "tightening" of deadlines for the first "run" has not been sufficient to effect changes in job priorities at any of the critical stages. Thus, comparison with figure 29 reveals identical decisions throughout, with no change whatsoever in the resulting schedule.

At the start of the second run (figure 37, tableau 8), the deadlines on the late jobs are tightened further, and the job slack rule is applied once again. Again, no change in priorities is effected (tableaus 9, 10, 11, 12.).

For the third run, deadlines are again tightened (tableau 14). This time, the result is to reverse the ordering of jobs at the second decision point (tableau 16). This change in and of itself leads to a new schedule, shown in tableau 21.

With original deadlines reimposed, all original latenesses (jobs 1, 3, and 5) are seen to have been eliminated.

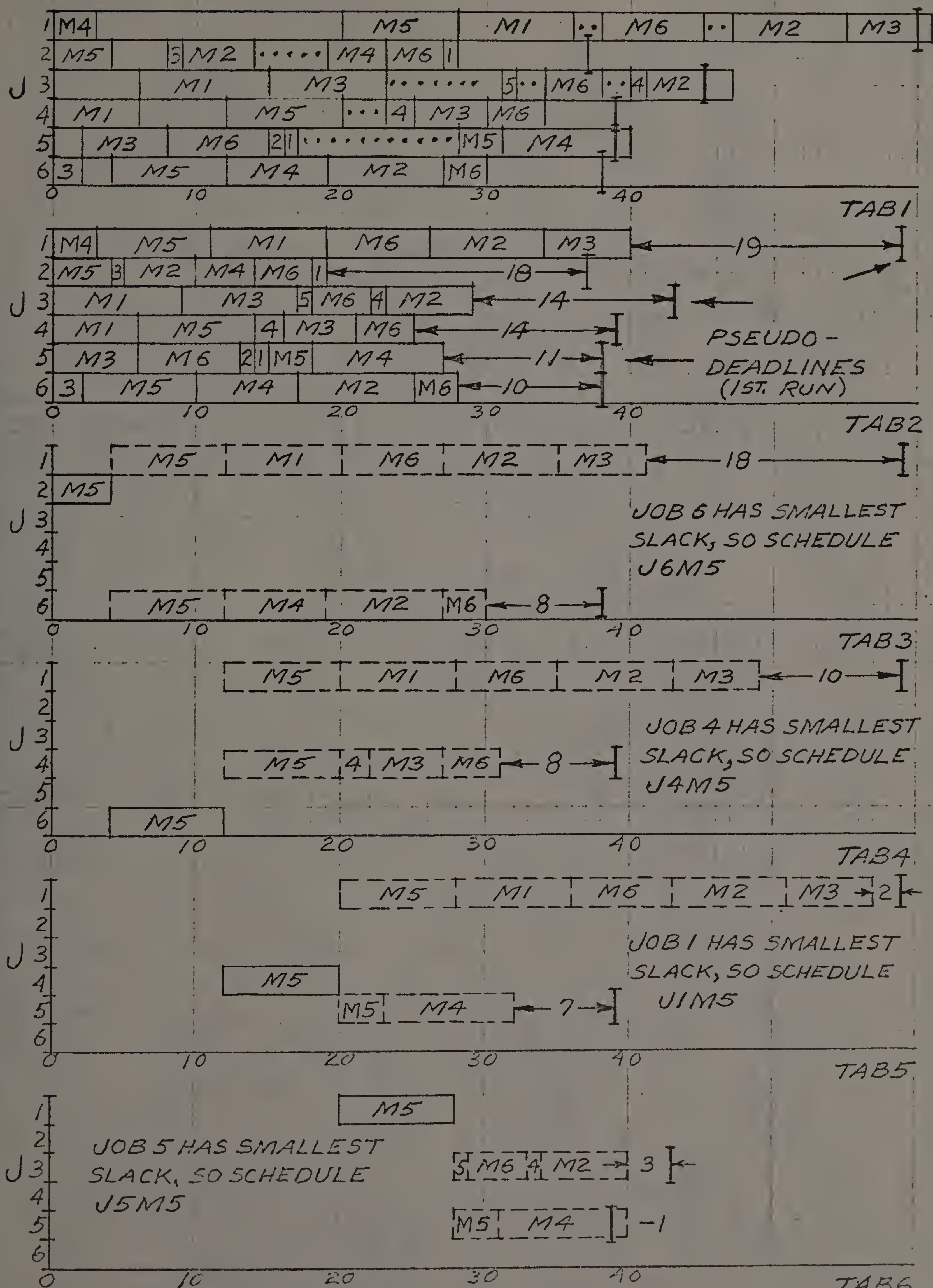
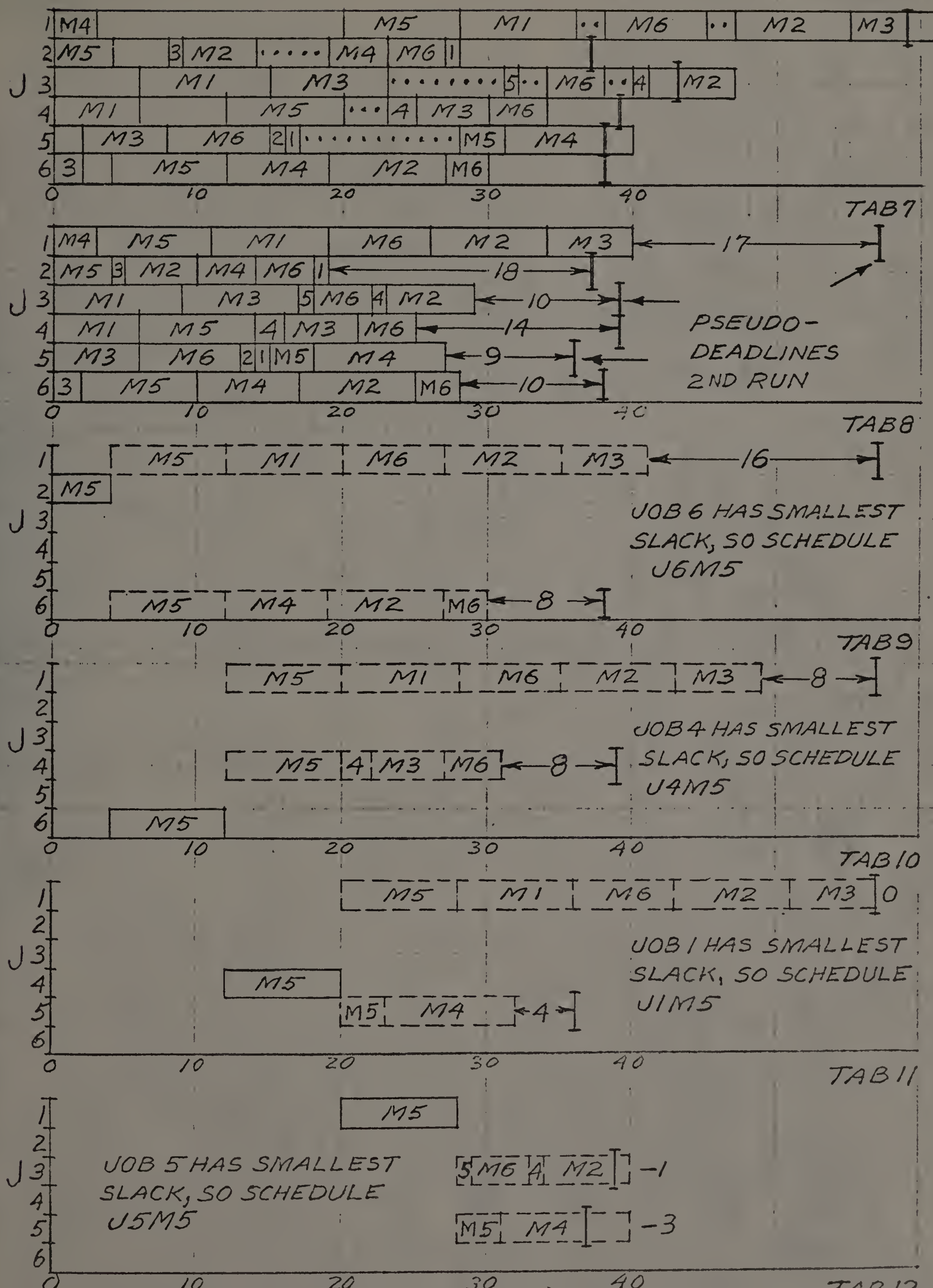

TAB1
TAB2
PSEUDO-DEADLINES (1ST. RUN)
19
18
14
14
11
10
TAB3
JOB 6 HAS SMALLEST SLACK, SO SCHEDULE J6M5
TAB4
JOB 4 HAS SMALLEST SLACK, SO SCHEDULE J4M5
TAB5
JOB 1 HAS SMALLEST SLACK, SO SCHEDULE J1M5
TAB6
JOB 5 HAS SMALLEST SLACK, SO SCHEDULE J5M5

FIG. 37



TAB7
PSEUDO-DEADLINES 2ND RUN
TAB8
JOB 6 HAS SMALLEST SLACK, SO SCHEDULE J6M5
TAB9
JOB 4 HAS SMALLEST SLACK, SO SCHEDULE J4M5
TAB10
JOB 1 HAS SMALLEST SLACK, SO SCHEDULE J1M5
TAB11
JOB 5 HAS SMALLEST SLACK, SO SCHEDULE J5M5
TAB12

FIG.37 (CONT)

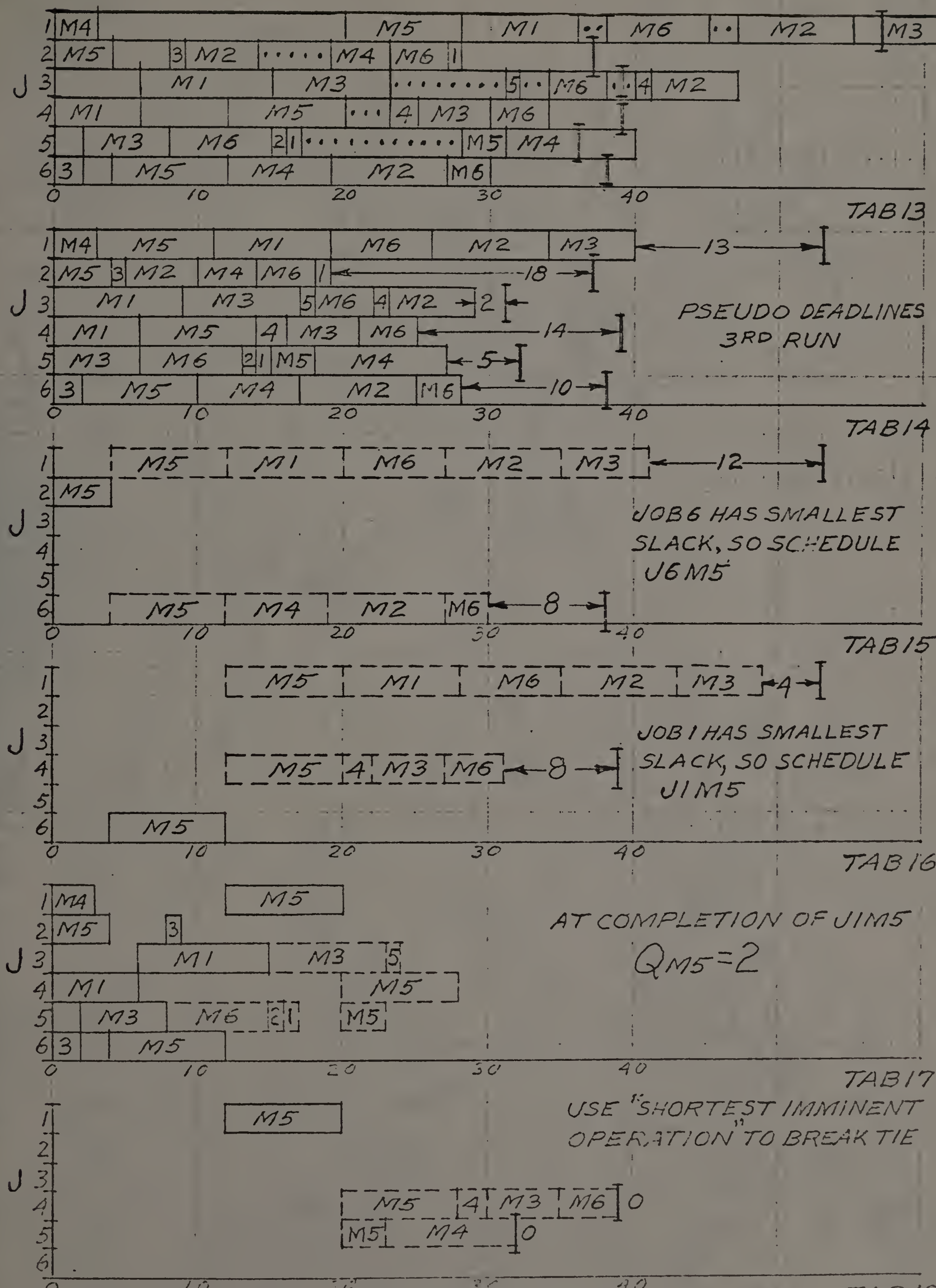

TAB 13
TAB 14
PSEUDO DEADLINES 3RD RUN
TAB 15
JOB 6 HAS SMALLEST SLACK, SO SCHEDULE J6 M5
TAB 16
JOB 1 HAS SMALLEST SLACK, SO SCHEDULE J1 M5
TAB 17
AT COMPLETION OF J1M5
QM5=2
TAB 18
USE "SHORTEST IMMINENT OPERATION" TO BREAK TIE

FIG. 37 (CONT)

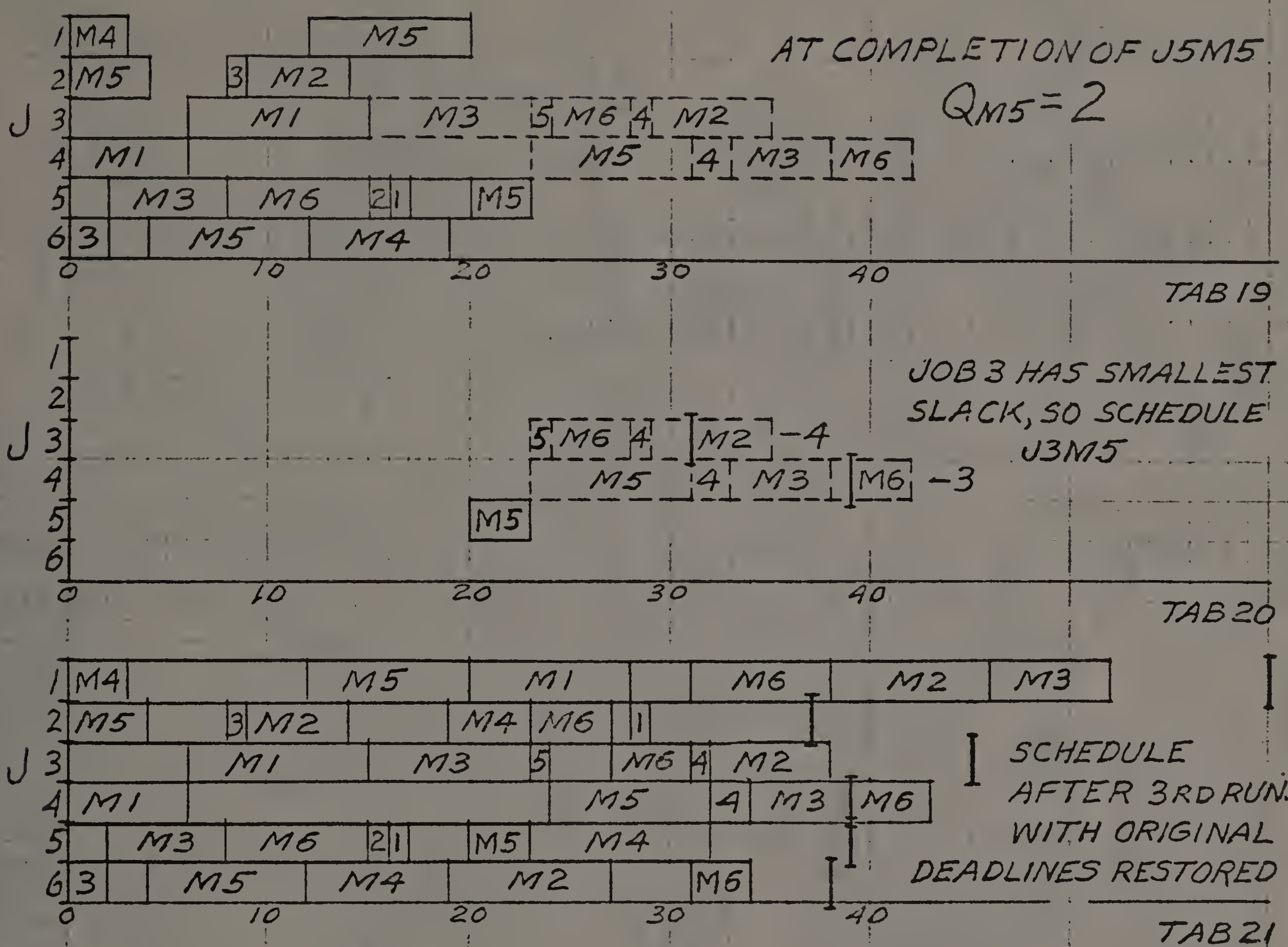

AT COMPLETION OF J5M5
$Q_{M5} = 2$
TAB 19
JOB 3 HAS SMALLEST SLACK, SO SCHEDULE J3M5
TAB 20
SCHEDULE AFTER 3RD RUN. WITH ORIGINAL DEADLINES RESTORED
TAB 21

FIG. 37 (CONT)

However, a new lateness of 4 time units now appears on job 4. (The amount of this lateness is exactly equal to the sum of the original latenesses.)

In order to effect any change in the original schedule, the artificial deadline rule had to be applied three times. In some problems, a change might have occurred on the first run, but it is not at all uncommon for a problem to require repeated applications. In any case, the need for reapplication is solely a function of problem parameters.

In the problem at hand, the new schedule represents, at best, a marginal improvement over the original one. Some additional improvement might be effected by repeating the procedure a fourth time, in which case only the deadline of job 4 would be tightened.

The artificial deadline rule is generally used as a substitute for the alternative assignment-idle time reduction rule-pair, although there is nothing that prevents both procedures from being tried in succession. Since the former method is apt to be operationally simpler, it should normally be tried first.

Conclusion

In this section, we have presented several heuristics which, when used in combination with basic priority rules, are likely to ameliorate the idle time and/or makespan characteristics of schedules. The intention was not to exhaust the list of possibilities; indeed, there exists a very large body of useful heuristics, of which the presently discussed set is only typical.

To establish some feeling for the enormity of this "body," it is only necessary to reflect on the tremendous range of logical thought processes that are used in solving problems of even moderate complexity. A motorist, for example, entering a rotary is, within a few brief seconds of time, forced to judge matters of velocity, acceleration, direction, location and distance. This calls for an extremely selective gathering of information, which is then processed according to such rules as "bear left to avoid running off the road" or"look in all directions to avoid getting hit." Adherence to these rules, as well as many others, makes it possible for the driver to successfully negotiate the rotary, even though such negotiation may not necessarily be of prototypic quality.

Each time a motorist enters a rotary, the parameters are quite different, yet he recognizes the nature of the decision-making process and applies logic accordingly. In the heuristic approach to scheduling, the procedure is very much the same. The parameters of every schedule are different, yet this does not prohibit the application of a host of logical rules which are used to successfully simplify, classify, and otherwise process the information.

A suitable heuristic rule is <u>any</u> which permits a satisfactory solution to a problem in which full assimilation of all information is either difficult or impossible. Much as the motorist is satisfied with his less-than-perfect negotiation of the rotary, so may the scheduler be satisfied with his less-than-perfect solution to the scheduling problem.

CHAPTER VI

SCHEDULING IN THE DYNAMIC ENVIRONMENT

Introduction

In previous chapters, a somewhat restrictive condition was placed on the scheduling problem, namely, that the set of jobs to be scheduled was completely known at the time the original schedule was devised and would not in any way be altered during the "life" of the schedule. Thus, once the schedule was generated, it was assumed that no orders would be cancelled, and that no new jobs would require scheduling until the original job set had been completed.

In practice, job sets rarely remain constant for fixed, predeterminate periods. More commonly, additions and deletions occur at random, thus invalidating the assumption that a carefully designed schedule need not be changed over a particular planning horizon.

Various factors dictate the need to remove uncompleted jobs from the job set. If a production contract is cancelled for any reason, the associated job (or jobs) are immediately taken ou- of production, in order to minimize losses. Since the elimination of such jobs opens up new capacity (i.e. creates idle time on facilities), it is generally possible to "compress" (and therefore improve) the original schedule by leftward shifting of operations. A general rescheduling of all jobs will probably not be required, although the merits of this conclusions should be evaluated within the context of each specific situation.

When new orders are received, the matter of scheduling may be approached in two ways. As a first possibility, the new orders may be ignored until the original set of orders is processed in its entirety. (This is consistent with "first-come, first served," the customer being informed that a back-log of orders awaits processing.) As a second possibility, however, it may be desirable to process a new order immediately, especially if the customer's business is held to be important and, additionally, he is unwilling to abide by the "first-come, first-served" rule.

The immediate (or near-immediate) processing of new orders may be achieved by pre-empting some currently-existing order of low priority. Another approach is simply to undertake a general rescheduling of all remaining operations, including those of the new job, and consistent with the guidelines suggested earlier in this paper. The first of these approaches is operationally simpler, but it tends to seriously delay the completion of the pre-empted job, since all its operations must now be scheduled last. In general, the latter approach is held to be superior because it results in a better "balancing" of priorities.

Deletion of Jobs

To illustrate job deletion, we refer back to the problem that was developed extensively in chapter three. It will be recalled that application of the shortest imminent operation rule led to the schedule of figure 18, tableau 17, repeated in figure 38, tableau 1. (There is nothing hallowed about this schedule; evidently, it is one of several that might have been chosen.)

Suppose now that at t=20, job 3 is suddenly cancelled. The schedule is therefore reduced to the configuration of figure 38, tableau 2. Since job 3 originally consumed some machine capacity, its omission permits the advancement of certain other jobs. The remaining operations of job 1, for example, may now be commenced one time unit earlier, and similar accommodations may be made for the operations of jobs 4 and 5, as illustrated in tableau 3.

The method illustrated is obviously intuitive, and can easily be applied for any number of deletions.

Addition of Jobs

When new jobs are added to the job set, a decision must be made regarding priorities. If it is determined that pre-empting should not occur, the original schedule remains un-disturbed until the original job set has been completed, whereupon the scheduling cycle begins afresh. On the other

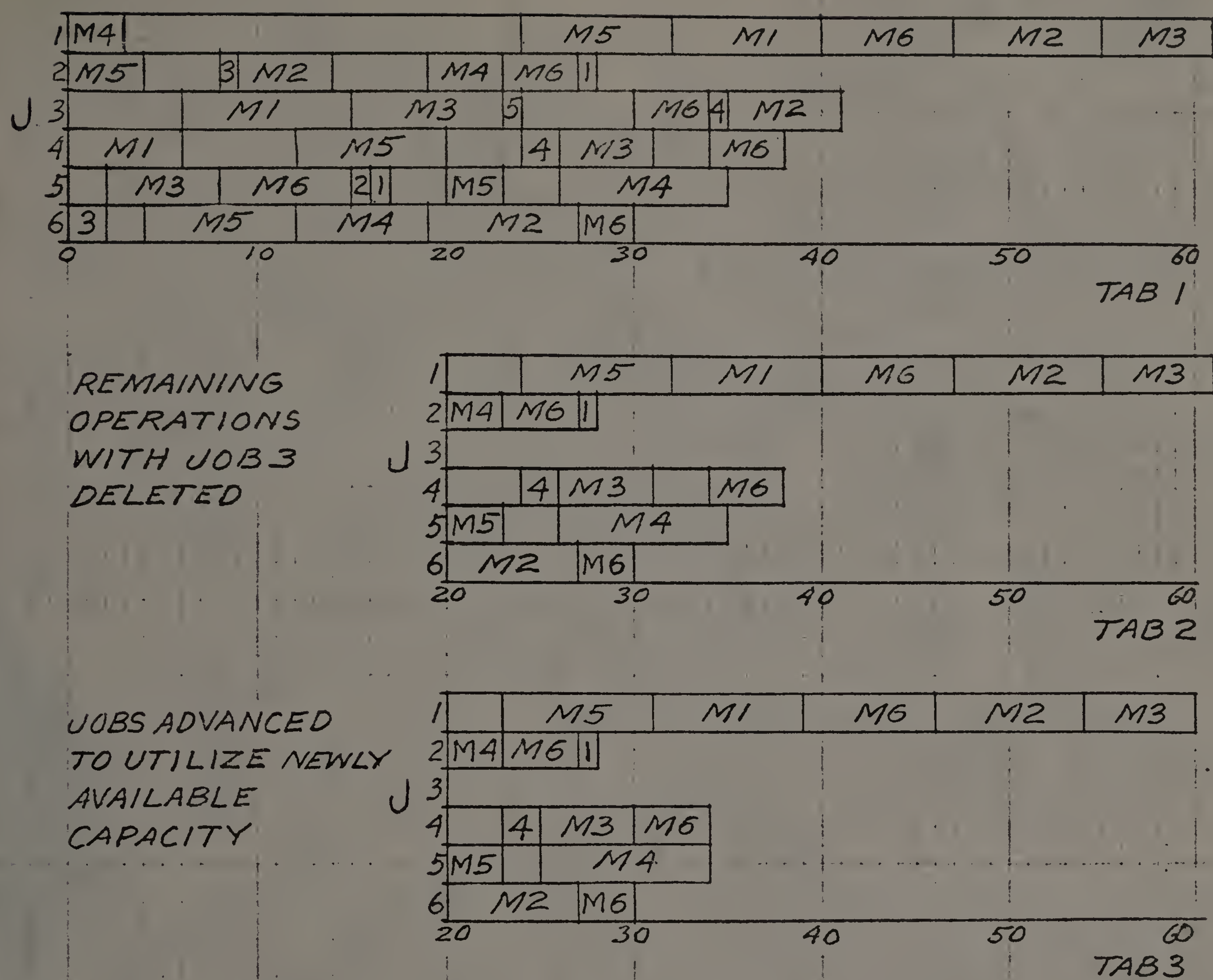

J
TAB 1
REMAINING OPERATIONS WITH JOB3 DELETED
TAB 2
JOBS ADVANCED TO UTILIZE NEWLY AVAILABLE CAPACITY
TAB3
M1
M2
M3
M4
M5
M6
0
10
20
30
40
50
60

FIG. 38

hand, if the new order cannot wait, a rescheduling of the original jobs may be undertaken. To illustrate, again consider figure 38, tableau 1, and assume that a new job, bearing the following properties, is to be commenced at t=20:

$$\left[F = \quad 76 \quad 74 \quad 71 \quad 73 \quad 72 \quad 75 \right]$$

Figure 39

$$\left[P = \quad 5 \quad 2 \quad 8 \quad 3 \quad 9 \quad 1 \right]$$

Figure 40

That is, job 7 is to be processed for 5 time units on machine 6, for 2 time units on machine 4, for 8 time units on machine 1, and so on.

The inclusion of this new order is probably best accomplished by laying out an entirely new schedule from t=20 on, assuming a new job set consisting of remaining operations of old jobs, plus the new job. This procedure is carried out in figure 41.

In figure 41, tableau 2, the part of the original schedule to the left of t=20 has been truncated. Additionally, all operations to the right of t=20, including those of the newly added job, have been left-justified as far as possible.

In figure 41, tableaus 3-12, the non-feasible schedule of tableau 2 is made feasible through application of the shortest imminent operation rule. The procedure used is precisely identical to the one developed in chapter three.

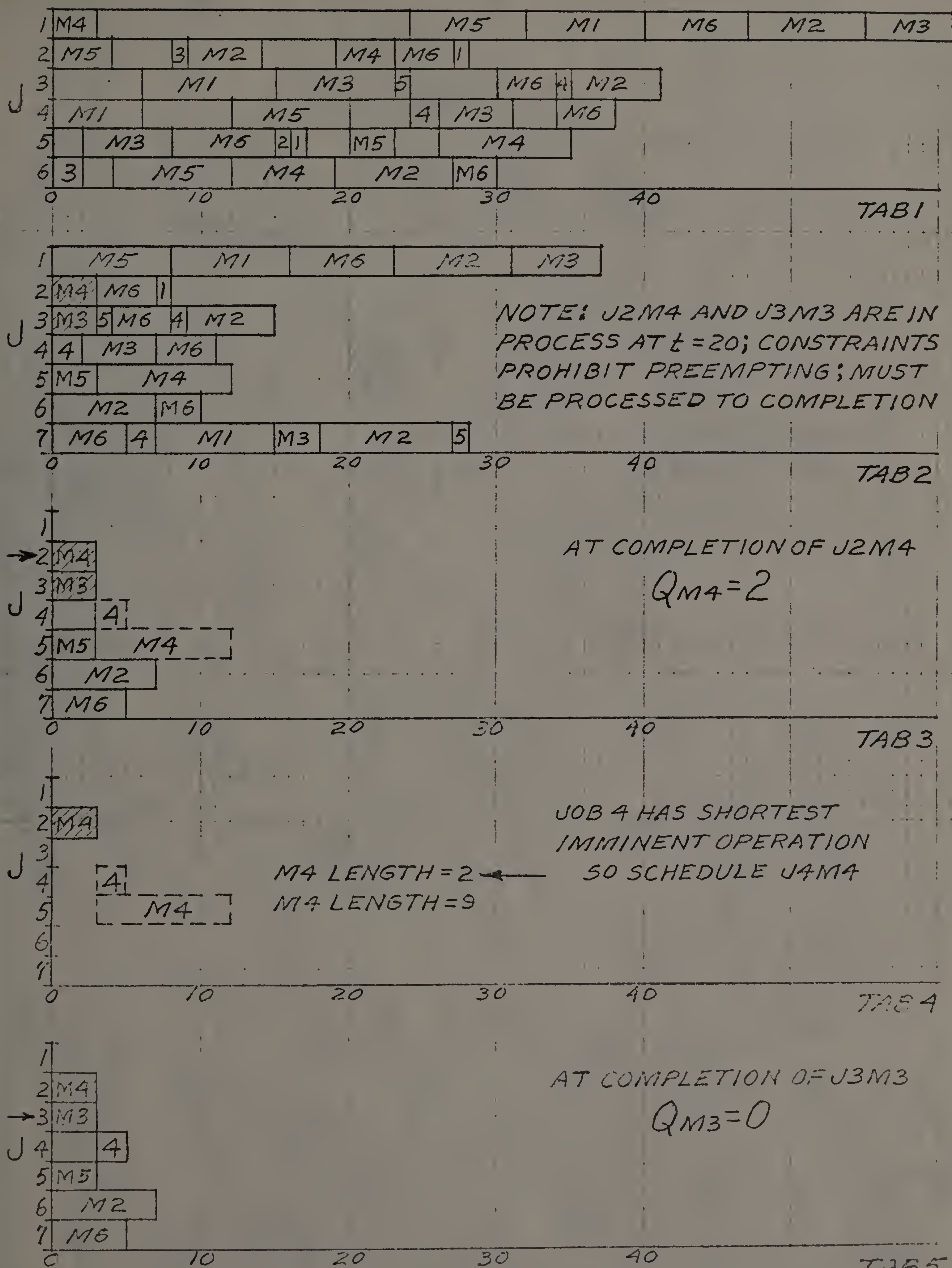

TAB 1
TAB 2
NOTE: J2M4 AND J3M3 ARE IN PROCESS AT t = 20; CONSTRAINTS PROHIBIT PREEMPTING; MUST BE PROCESSED TO COMPLETION
TAB 3
AT COMPLETION OF J2M4
$Q_{M4} = 2$
TAB 4
JOB 4 HAS SHORTEST IMMINENT OPERATION SO SCHEDULE J4M4
M4 LENGTH = 2
M4 LENGTH = 9
TAB 5
AT COMPLETION OF J3M3
$Q_{M3} = 0$

FIG. 41

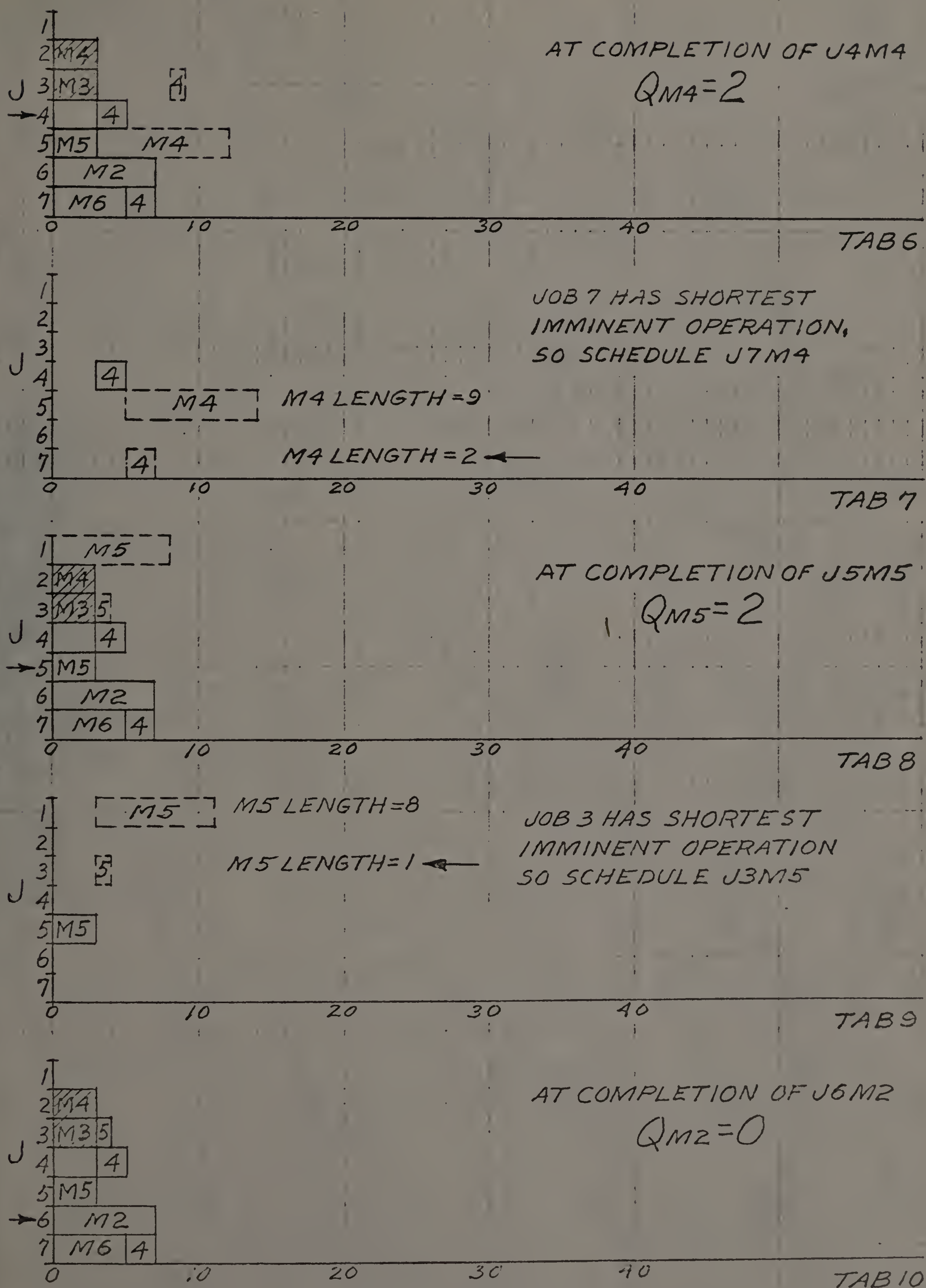

AT COMPLETION OF J4M4
QM4=2
TAB 6
JOB 7 HAS SHORTEST
IMMINENT OPERATION,
SO SCHEDULE J7M4
M4 LENGTH=9
M4 LENGTH=2
TAB 7
AT COMPLETION OF J5M5
QM5=2
TAB 8
M5 LENGTH=8
M5 LENGTH=1
JOB 3 HAS SHORTEST
IMMINENT OPERATION
SO SCHEDULE J3M5
TAB 9
AT COMPLETION OF J6M2
QM2=0
TAB 10

FIG. 41 (CONT)

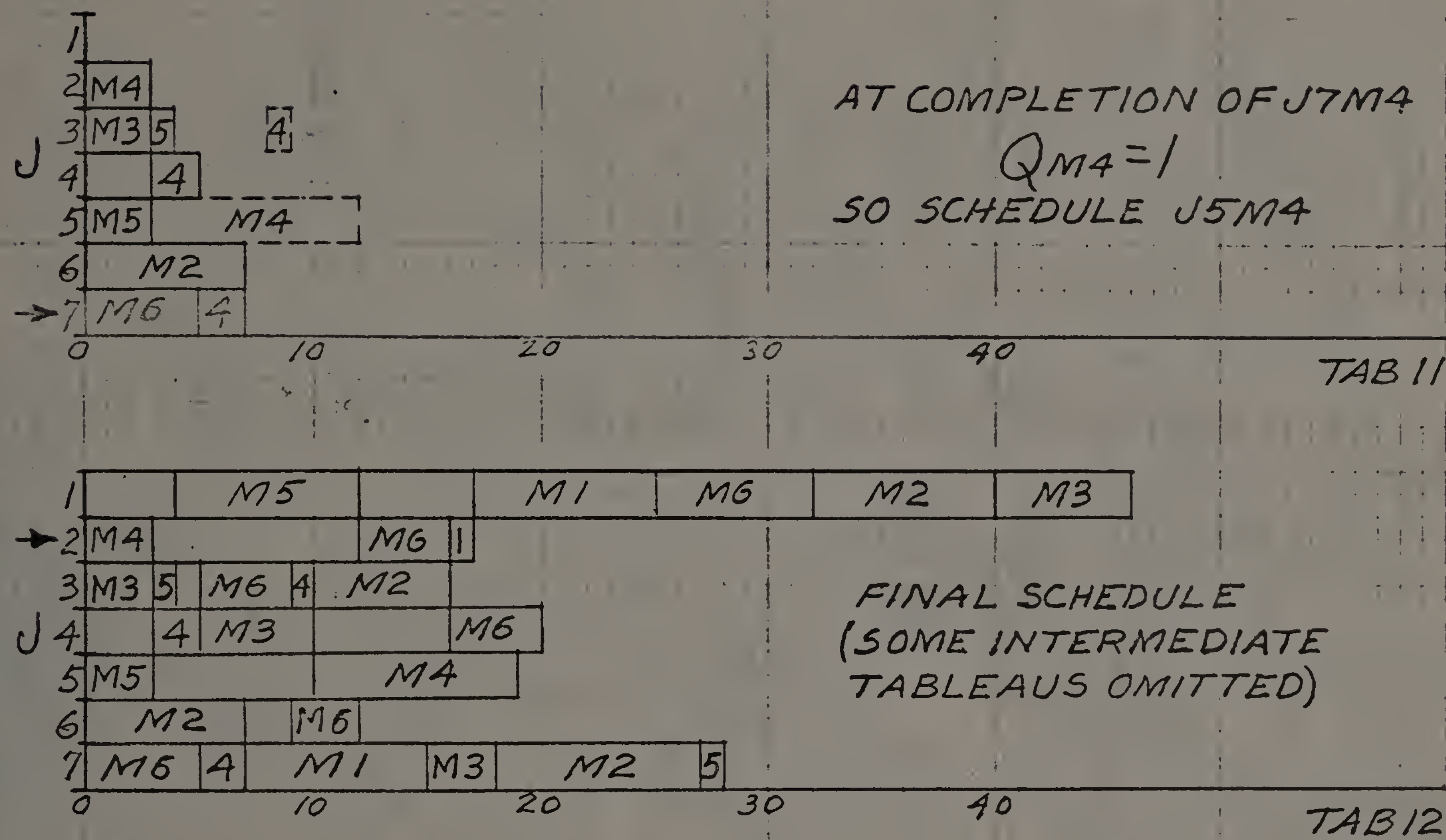

AT COMPLETION OF J7M4
QM4=1
SO SCHEDULE J5M4
TAB 11
FINAL SCHEDULE
(SOME INTERMEDIATE
TABLEAUS OMITTED)
TAB 12

FIG. 41 (CONT)

(Note that any operations which are in process at the start of the new schedule cannot be interrupted, according to the non-pre-emptive priority constraint.)

The example illustrates job addition, consistent with application of the shortest imminent operation rule. Any other rule might equally well have been applied, with or without the use of supplementary heuristics.

CHAPTER VII

THE FLOWSHOP PROBLEM

Introduction

In the previous chapters, we considered procedures for obtaining satisfactory solutions to the general job shop problem. The problem, it will be recalled, could not be treated by purely formulary means, and the alternative was to apply basic priority rules in conjunction with certain supplemental heuristics.

Let us once again examine the combinatorial aspects of the general job shop case. If each of n jobs require processing once and only once on each of m machines, the total number of feasible schedules is $(n!)^m$. For example, in the case of two jobs and three machines, jobs can be dispatched to each machine in two possible sequences, namely, job 1 first and job 2 second, or job 2 first and job 1 second. Since there are three machines, and each job requires processing on each machine, this results in a total of (2)(2)(2) $=(2!)^3=8$ possible ways in which jobs can be dispatched to machines, that is, in a total of eight distinct feasible schedules.

Suppose now, however, that we introduce an additional constraint, namely that the ordering of operations is the same for all jobs. Under these conditions, the once-formidable job-shop problem degenerates into a relatively simple special case known as the flowshop problem.

In a paper dealing primarily with general solutions, there is considerable justification for devoting attention to the flowshop problem. Flowshop situations (or approximations) occur very frequently in practice, since operational orderings tend to be dictated by technological considerations rather than by the peculiar characteristics of individual jobs. For example, the manufacture of a precision machine part might begin with the production of a rough casting, subsequently to be ground, polished, drilled and tapped, in that order. Other job orders might call for machine parts of various sizes, shapes and tolerances, but, in general, one would expect operations ordering to be reasonably consistent. (In certain instances, there is simply no choice. Drilling could occur prior to grinding or polishing, but tapping could not possibly occur prior to drilling!)

An "approximation" to a flowshop problem is defined as one whose jobs share a common operational sequencing with one or two exceptions. Under such circumstances, it is quite appropriate to treat the situation as a "pure" flowshop, accommodating deviant operations consistent with the rules developed in the earlier sections of this paper.

To illustrate the way in which the flowshop assumption simplifies a given problem, consider the sample problem of chapter two, but modify it so that

$$F = \begin{bmatrix} 13 & 11 & 12 \\ 23 & 21 & 22 \end{bmatrix}$$

Figure 42

That is, two jobs, each with three operations, both require processing, first on machine 3, second on machine 1 and third on machine 2. (Recall that previously, the order of operations was different for each job.

As before,

$$P = \begin{bmatrix} 5 & 2 & 3 \\ 1 & 4 & 3 \end{bmatrix}$$

Figure 43

Thus, job 1 is processed for 5 minutes on machine 3, followed by 2 minutes on machine 1, and, finally, for 3 minutes on machine 2. In job 2, the order of operations is now identical to job 1, but the processing times involved are 1, 4 and 3 minutes, respectively.

Under these conditions, all feasible schedules are enumerated in figure 44, tableaus 1a to 8a. Clearly there are still eight distinct schedules, corresponding to the eight possible ways in which jobs can be dispatched to machines.

At this point one might justifiably question the value of imposing a uniform operations-ordering restriction, since no particular benefits, in terms of problem simplification, seem to accrue. Closer inspection, however, reveals a peculiar characteristic of the more desirable schedules (i.e.

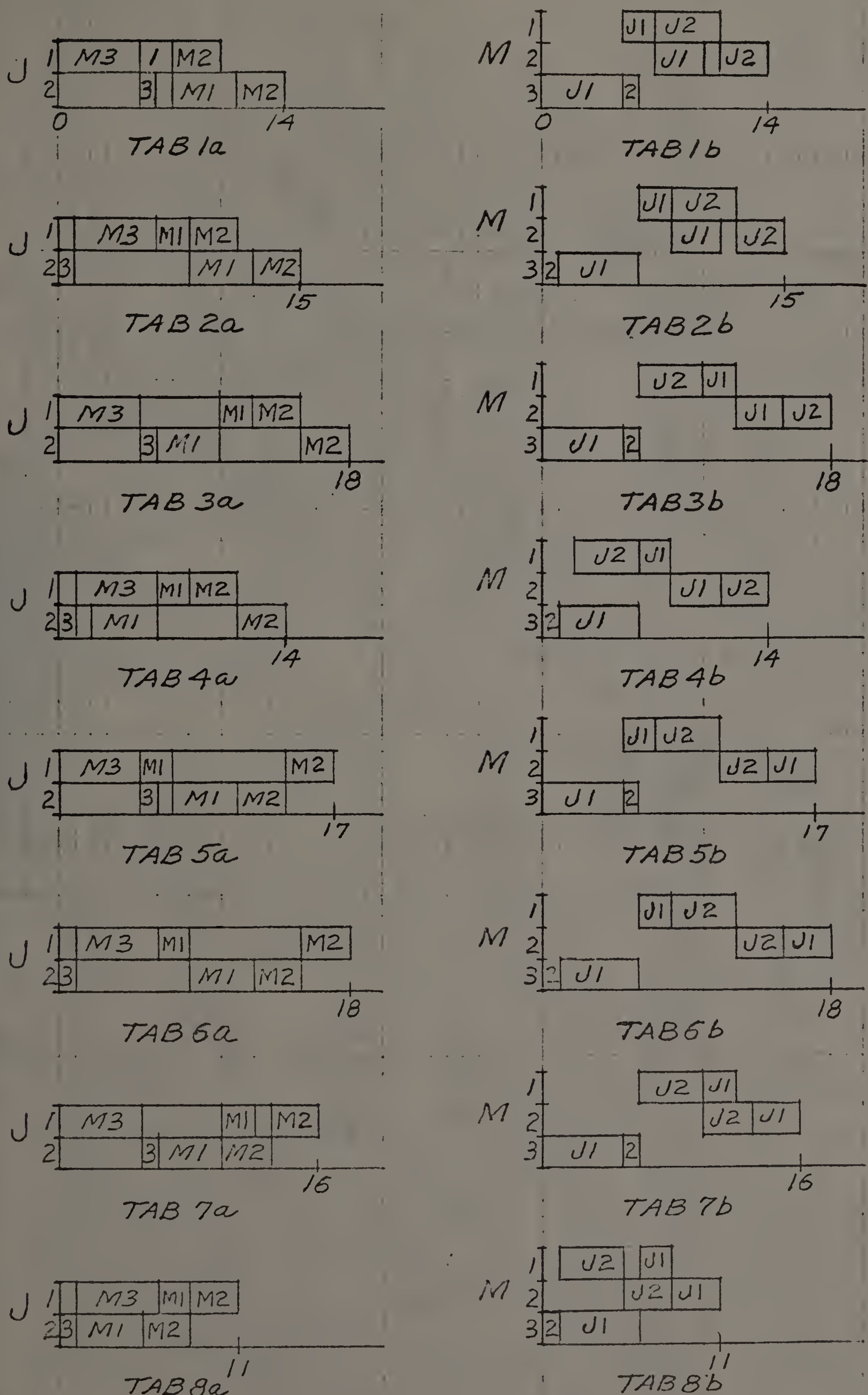

TAB 1a
TAB 1b
TAB 2a
TAB 2b
TAB 3a
TAB 3b
TAB 4a
TAB 4b
TAB 5a
TAB 5b
TAB 6a
TAB 6b
TAB 7a
TAB 7b
TAB 8a
TAB 8b

FIG. 44

those with the shorter makespans) that has potential usefulness as a basis for the formulation of a general dispatching rule.

Figure 44a lists schedule makespans in descending order of magnitude. Referral to figure 44 immediately reveals that

SCHEDULE	MAKESPAN
3a	18
6a	18
5a	17
7a	16
2a	15
4a	14
1a	14
8a	11

Figure 44a

the schedules with the shortest makespans (i.e. 1a and 8a) are those in which all jobs have processing priorities that remain fixed from operation to operation. On the other hand, the schedules with the longer makespans are those in which the priorities of jobs vary. To illustrate the former point, notice that schedule 1a gives first priority to job 1 at every operation, while schedule 8a consistently gives first priority to job 2. (In the first case, J1M3 precedes J2M3, J1M1 precedes J2M1, and J1M2 precedes J2M2; in the second, J2M3 precedes J1M3, J2M1 precedes J1M1, and J2M2 precedes

J1M2.) In support of the latter, note the mixed processing priorities of schedules 2a-7a. (In 6a, for example, J2M3 precedes J1M3, and J2M2 precedes J1M2; however, J1M1 precedes J2M1.)

If it is, in fact, appropriate to generalize, we conclude that an optimal (or near optimal) flowshop schedule results from the judicious sequencing of jobs rather than operations within jobs. According to this line of reasoning, the following rules would seem appropriate: To obtain an optimal schedule,

(1) Determine an optimal set of scheduling priorities for jobs (i.e. determine which job should be scheduled first, which should be scheduled second, and so on.)

(2) Schedule each machine according to the priorities established in (1). (Consider machines in the order specified by the facility-ordering matrix.)

Clearly, all that remains is to determine an efficient means for generating an optimal (or near optimal) set of job priorities. In the problem heretofore considered, simple enumeration could be used to determine this set, since only two jobs were involved. (Comparison between the makespans of schedules 1a and 8a leads to the conclusion that job 2 should be afforded first priority.) On the other hand, enumeration would be impractical in problems of any reasonable size. (The reader should verify that an n-job flowshop problem has n! feasible solutions remaining even after the elimination of schedules involving mixed job-processing priorities!)

While somewhat less staggering than that of the general job shop case, the simplified flowshop problem still presents an extremely difficult combinatorial exercise. Fortunately, however, the solution can be approached somewhat more directly than in the job-shop case. Two relatively simple heuristic techniques have been devised which, on the basis of the empirical evidence, appear to have very high likelihoods of generating optimal or very-near optimal solutions to the flowshop problem. As will be shown, both techniques make a common assumption, namely, that the efficiency of schedules depends on the extent to which shortest operations are scheduled first.

To expand on this latter point, consider the fixed-processing-priority schedules 1a and 8a, but modify them so that machines are now designated along the vertical axis, and blocks contain job information. (To avoid confusion, the machine designations should always be sequenced in an order consistent with the facility-ordering matrix.) Schedules 1a and 8a would be modified thus:

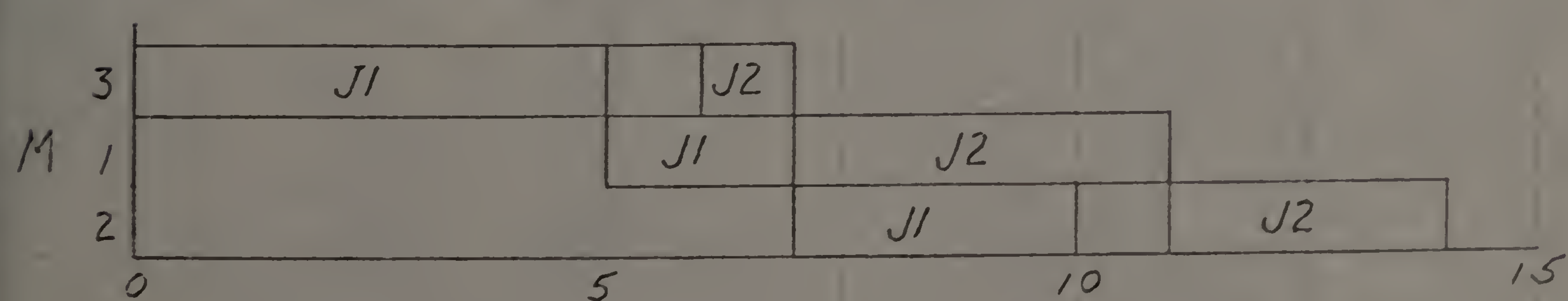


Figure 45

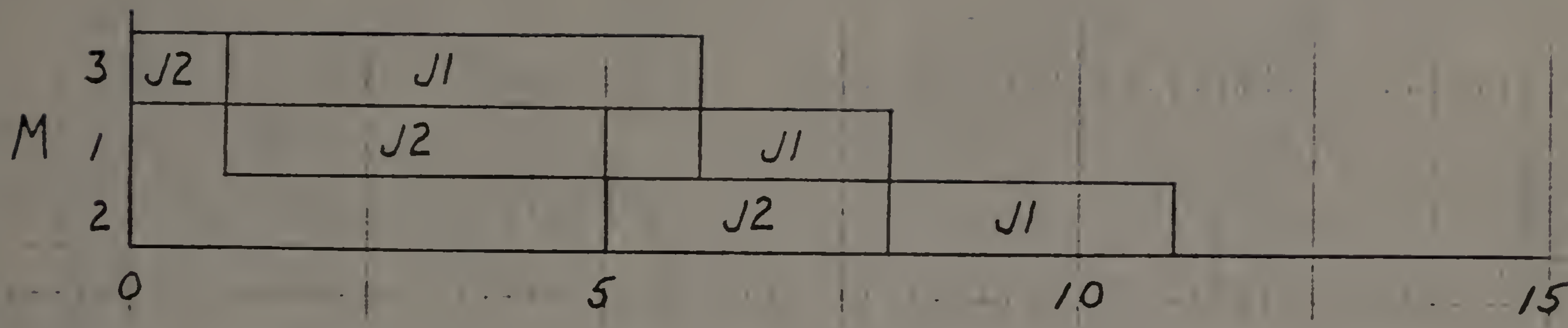


Figure 46

The distinguishing features of the optimal schedule are now abundantly clear. Job 1 has a very long beginning operation (J1M3), which, in figure 45, prevents the start of job 2 for a very long time. Hence the optimal schedule (figure 46) is the one in which job 1 is scheduled last.

The results of the example can be generalized: Overall total processing time (makespan) tends to be reduced if jobs with early short operations are scheduled first. This procedure tends to aggregate all short operations near the beginning of the schedule, thus reducing the likelihood of lengthy bottlenecks at early stages. The validity of the procedure improves to the extent that operational times within jobs tend to increase or decrease in monotonic fashion (i.e. any job can be characterized as having "the bulk" of its short operations either "near the beginning" or "near the end" as opposed to having short and long operations mixed throughout in no discernable pattern.)

In a problem involving many jobs, simple inspection is frequently not sufficient to discriminate the best schedule. The difficulty is traceable to the absence of any specific criteria which might be used to govern the ranking of jobs. On the basis of the discussion thus far, it would be impossible, for example, to determine which of two jobs should be given higher priority: one with many moderately short operations at the beginning, or one with a single very short operation followed be several moderately long ones.

Palmer Slope Method[1]

As a means toward resolving issues of this type, Palmer suggests the computation of a numerical "slope index," which assigns weighting factors to operational times for each job. These weighting factors are arranged in a continuum such that early operations receive heavy negative weightings, central operations receive small negative or positive or zero weightings, and terminal operations receive heavy positive weightings. The suggested slope index is given as:

$$S_j = -\frac{m-1}{2}t_{j1}-\frac{m-3}{2}t_{j2}\cdots+\frac{m-3}{2}t_{j(m-1)}+\frac{m-1}{2}t_{jm}.$$

where j refers to the job in question, m is the number of machines in the problem, and t_{ji} is the processing (operational) time required by the jth job on the ith machine.

The logic behind the "slope index" approach is not difficult to discern. Jobs with operational times that are monotonic decreasing (or nearly so) throughout will have negative slope indices, since the longer operations at the start are assigned heavy negative weightings. On the other hand, jobs with operational times that are monotonic increasing (or nearly so) throughout will have positive slope indices, since the longer operations near the end are assigned heavy positive weightings. The greater the tendency to cluster very long jobs at the beginning, the more negative the slope index; the greater the tendency to cluster very short jobs at the beginning, the more positive the slope index.

If scheduling of short operations first is held to be a desirable condition (see p.139), then jobs should be scheduled according to slope index positivity. Specifically, the job having the most positive slope index should be scheduled first, the job having the second most positive index should be scheduled second, and so on. This procedure should result in the clustering of all short operations at the beginning of the schedule.

It should be emphasized, partly in the way of reiteration, that precision of results depends largely on the extent to which jobs can be characterized as having operations whose times increase or decrease in approximately monotonic fashion. In most problems, there are generally at least a few jobs with reasonable operational configurations, and these tend to represent extremes, in terms of ranking. The remaining jobs tend to be assigned central rankings, and it is here that inaccuracies are most likely to occur. However, in many cases, it will turn out that only one or two priority designations are open to serious question. Under these conditions, the Palmer method would still be expected to produce a nearly-optimal solution to the flowshop problem.

Examples. The Palmer method is now applied to two sample problems. The first is the problem at hand in which:

$$F = \begin{bmatrix} 13 & 11 & 12 \\ 23 & 21 & 22 \end{bmatrix}$$

Figure 47

and

$$P = \begin{bmatrix} 5 & 2 & 3 \\ 1 & 4 & 3 \end{bmatrix}$$

Figure 48

Applying the formulas, we obtain:

$$S_1 = -\frac{3-1}{2}(5) \pm \frac{3-3}{2}(2) + \frac{3-1}{2}(3)$$

$$S_2 = -\frac{3-1}{2}(1) \pm \frac{3-3}{2}(4) + \frac{3-1}{2}(3)$$

$$S_1 = -5+3 = -2$$

$$S_2 = -1+3 = +2$$

S_2 is the most positive, so jobs are processed in the order (2 1). The resulting schedule is evidently the same as the one in figure 44, tableau 8a.

It is instructive to note that the facility-ordering matrix does not enter into the solution at all. This is because it is the sequence of operational times that is important here; the sequence of machines on which processing is to occur is of no consequence.

The second example involves the job-shop problem dealt with earlier in the dissertation. Recall that

$$F = \begin{bmatrix} 14 & 15 & 11 & 16 & 12 & 13 \\ 25 & 23 & 22 & 24 & 26 & 21 \\ 31 & 33 & 35 & 36 & 34 & 32 \\ 42 & 41 & 45 & 44 & 43 & 46 \\ 53 & 56 & 52 & 51 & 55 & 54 \\ 63 & 65 & 64 & 62 & 66 & 61 \end{bmatrix}$$

Figure 49

and

$$P = \begin{bmatrix} 3 & 8 & 8 & 7 & 8 & 6 \\ 4 & 1 & 5 & 4 & 4 & 1 \\ 9 & 8 & 1 & 4 & 1 & 6 \\ 0 & 6 & 8 & 2 & 5 & 4 \\ 6 & 7 & 1 & 1 & 3 & 9 \\ 2 & 8 & 7 & 8 & 3 & 0 \end{bmatrix}$$

Figure 50

To transform the job-shop problem into a flow-shop case, we arbitrarily pick any facility-ordering sequence and apply it to all jobs. Selecting (1 3 5 6 4 2), the facility-ordering matrix becomes:

$$F = \begin{bmatrix} 11 & 13 & 15 & 16 & 14 & 12 \\ 21 & 23 & 25 & 26 & 24 & 22 \\ 31 & 33 & 35 & 36 & 34 & 32 \\ 41 & 43 & 45 & 46 & 44 & 42 \\ 51 & 53 & 55 & 56 & 54 & 52 \\ 61 & 63 & 65 & 66 & 64 & 62 \end{bmatrix}$$

Figure 51

The processing-time matrix is thus modified to:

$$P = \begin{bmatrix} 8 & 6 & 8 & 7 & 3 & 8 \\ 1 & 1 & 4 & 4 & 4 & 5 \\ 9 & 8 & 1 & 4 & 1 & 6 \\ 6 & 5 & 8 & 4 & 2 & 0 \\ 1 & 6 & 3 & 7 & 9 & 1 \\ 0 & 2 & 8 & 3 & 7 & 8 \end{bmatrix}$$

Figure 52

Applying the slope index formulas, we obtain:

$$S_1 = -\frac{6-1}{2}(8) - \frac{6-3}{2}(6) - \frac{6-5}{2}(8) + \frac{6-5}{2}(7) + \frac{6-3}{2}(3) + \frac{6-1}{2}(8) = -5.0$$

$$S_2 = -\frac{6-1}{2}(1) - \frac{6-3}{2}(1) - \frac{6-5}{2}(4) + \frac{6-5}{2}(4) + \frac{6-3}{2}(4) + \frac{6-1}{2}(5) = +14.5$$

$$S_3 = -\frac{6-1}{2}(9) - \frac{6-3}{2}(8) - \frac{6-5}{2}(1) + \frac{6-5}{2}(4) + \frac{6-3}{2}(1) + \frac{6-1}{2}(6) = -16.5$$

$$S_4 = -\frac{6-1}{2}(6) - \frac{6-3}{2}(5) - \frac{6-5}{2}(8) + \frac{6-5}{2}(4) + \frac{6-3}{2}(2) + \frac{6-1}{2}(0) = -21.5$$

$$S_5 = -\frac{6-1}{2}(1) - \frac{6-3}{2}(6) - \frac{6-5}{2}(3) + \frac{6-5}{2}(7) + \frac{6-3}{2}(9) + \frac{6-1}{2}(1) = +6.5$$

$$S_6 = -\frac{6-1}{2}(0) - \frac{6-3}{2}(2) - \frac{6-5}{2}(8) + \frac{6-5}{2}(3) + \frac{6-3}{2}(7) + \frac{6-1}{2}(8) = +25.0$$

Jobs are therefore processed in the order (6 2 5 1 3 4).

The resultant schedule is shown below:

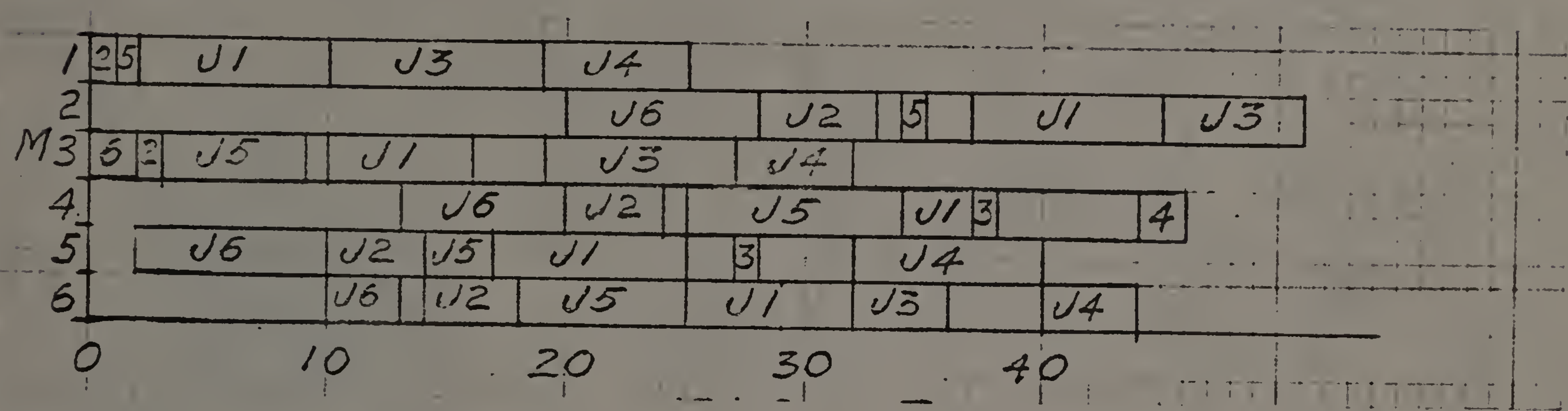


Figure 53

As noted earlier, jobs whose operational lengths are in no particular order tend to end up near the middle of the ranking group. (This is because their slope indices tend to approach zero.) Since the slope method is notoriously ineffective in generating meaningful rankings for such jobs, it is sometimes useful to arbitrarily juxtapose the middlemost rankings of the ranking group, and test the resulting schedule.

In the problem at hand, the middlemost rankings are 5 and 1. When these are juxtaposed, the ranking group becomes (6 2 1 5 3 4). The alternative schedule is:

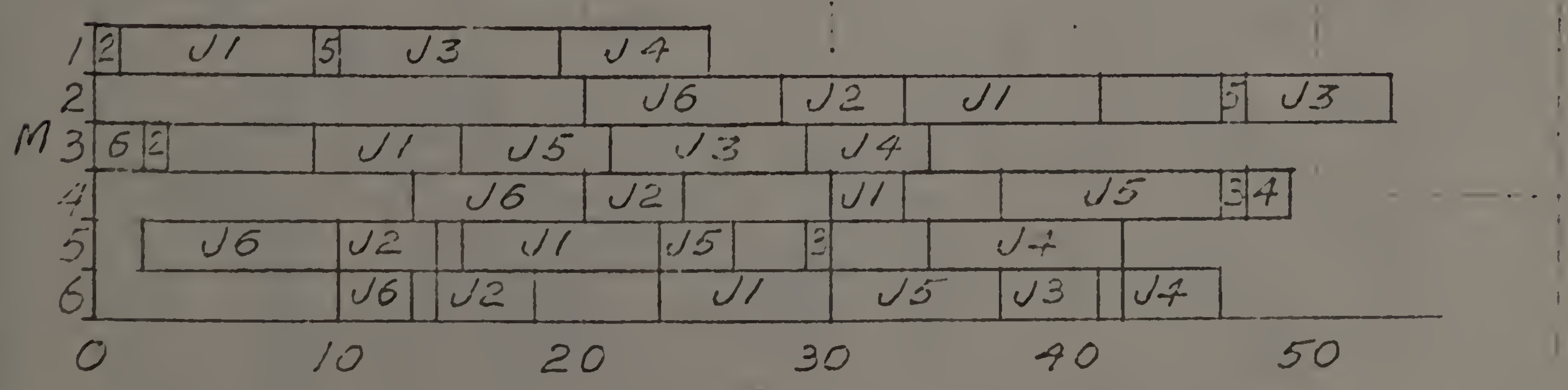


Figure 54

Since this schedule is two time units longer than the original one, it is rejected, and the original ranking set is restored.

There is nothing that prevents the scheduler from examining other permutations of the central elements of the ranking set. He might, for example, being of a highly suspicious nature, wish to examine permutations of the middlemost four elements, rather than just the two. This approach would only be limited by the difficulties involved in enumerating large numbers of schedules - the very same difficulties that led to utilization of the slope method in the first place!

In conclusion, the Palmer slope method represents a logical and straight-forward way to obtain a near-optimal solution to the flowshop problem. An alternative to the slope method will now be considered.

Campbell-Dudek-Smith Algorithm[2]

This method involves the decomposition of a problem involving several machines into multiple two-machine problems, each of which can be solved exactly. A limited number of job sequences are generated, and the sequence that results in shortest makespan is found by direct evaluation.

The Campbell-Dudek-Smith method utilizes, as its basis, the two-machine procedure of S.M. Johnson.[3] The two-machine case is trivial and has an exact solution because the two-machine restriction forces a precisely monotonic pattern on processing times. Thus, jobs can be unambiguously ranked with respect to their tendencies to have short first operations and long second ones, or vice versa. Specifically, these "tendencies" can be determined by computing the ratio of first operation length to second operation length for each job. If the scheduling of short operations first is held to be a desirable condition, the job with the smallest ratio (i.e. shortest first operation relative to second) should be scheduled first, the job with the second smallest ratio should be scheduled second, and so on.

The original statement of the Johnson procedure tends to obscure its underlying logic. The steps are best restated as follows:

(1) For each machine-pair (job) in the processing-time matrix, compute the ratio of the first operation time to the second.

(2) Establish job rankings based on the ratios. Schedule the job with the smallest ratio first, the job with the second smallest ratio second, and so on.

To illustrate the procedure, consider a two-machine problem in which

$$P = \begin{bmatrix} 6 & 3 \\ 1 & 4 \\ 8 & 1 \\ 5 & 2 \\ 6 & 9 \\ 2 & 7 \end{bmatrix}$$

Figure 55

The ratios for each job are computed as follows:

J1: 6/3 = 2

J2: 1/4 = .25

J3: 8/1 = 8

J4: 5/2 = 2.5

J5: 6/9 = .667

J6: 2/7 = .25

The ranking which assigns highest priorities to the jobs with the smallest ratios is (2 6 5 1 4 3). The associated Gantt Chart, with a makespan of 29 time units, is shown in figure 56.

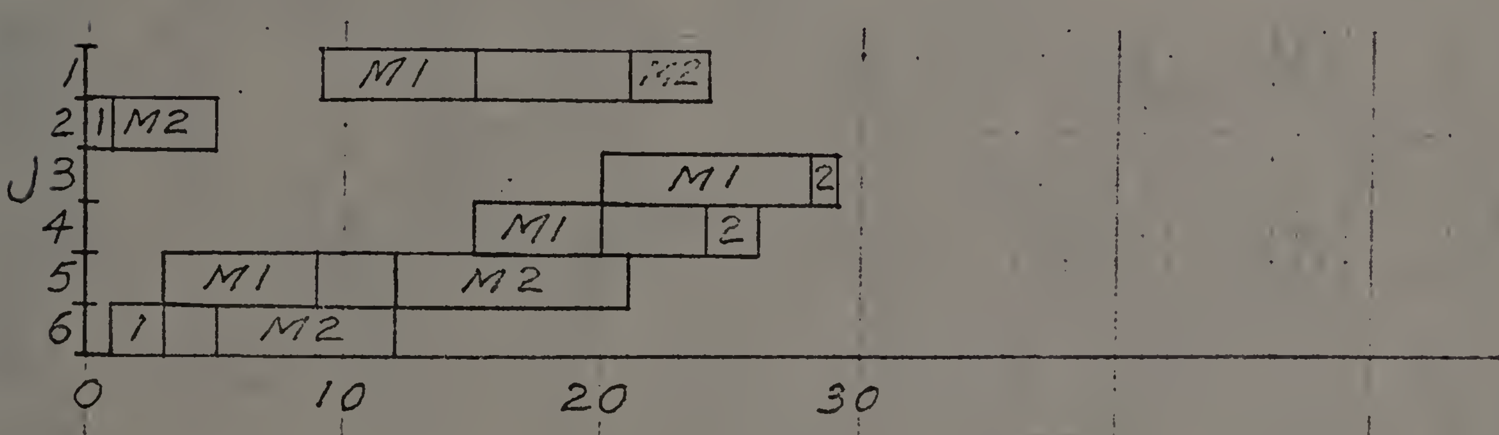


Figure 56

Having discussed the procedure for exact solution of two-machine problems, we are now ready to discuss the Campbell-Dudek-Smith method in detail. For a problem involving n jobs and m machines, the algorithm is as follows:

(1) Construct the first two-machine problem from the 1st and mth columns of the processing time matrix P. Solve this problem using the suggested procedure.

(2) Construct the second two-machine problem using the <u>sum</u> of columns 1 through 2 and the <u>sum</u> of columns m-1 through m. Solve this problem as in (1).

.
.
.

(3) Construct the (m-1)st two-machine problem using the <u>sum</u> of columns 1 through m-1 and the <u>sum</u> of columns m-(m-1) through m. Solve as previously.

(4) Having solved m-1 two-machine problems, and having generated m-1 feasible ranking sets, select the most efficient job sequence by direct evaluation.

<u>Examples</u>. To facilitate examination of the logic behind the foregoing procedure, we revert to the previously considered example wherein

$$
P = \begin{bmatrix}
1 & 2 & . & . & . & .m-1 & m \\
8 & 6 & 8 & 7 & & 3 & 8 \\
1 & 1 & 4 & 4 & & 4 & 5 \\
9 & 8 & 1 & 4 & & 1 & 6 \\
6 & 5 & 8 & 4 & & 2 & 0 \\
1 & 6 & 3 & 7 & & 9 & 1 \\
0 & 2 & 8 & 3 & & 7 & 8
\end{bmatrix}
$$

Figure 57

The m-1 two-machine problems to be solved are evidently

$$P_1 = \begin{bmatrix} 8 & 8 \\ 1 & 5 \\ 9 & 6 \\ 6 & 0 \\ 1 & 1 \\ 0 & 8 \end{bmatrix} \qquad P_4 = \begin{bmatrix} 29 & 26 \\ 10 & 17 \\ 22 & 12 \\ 23 & 14 \\ 17 & 20 \\ 13 & 26 \end{bmatrix}$$

$$P_2 = \begin{bmatrix} 14 & 11 \\ 2 & 9 \\ 17 & 7 \\ 11 & 2 \\ 7 & 10 \\ 2 & 15 \end{bmatrix} \qquad P_5 = \begin{bmatrix} 32 & 32 \\ 14 & 18 \\ 23 & 20 \\ 25 & 19 \\ 26 & 26 \\ 20 & 28 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} 22 & 18 \\ 6 & 13 \\ 18 & 11 \\ 19 & 6 \\ 10 & 17 \\ 10 & 18 \end{bmatrix}$$

The ratios for each problem are

$$P_1 = \begin{cases} 8/8 = 1 \\ 1/5 = .2 \\ 9/6 = 1.5 \\ 6/0 = \infty \\ 1/1 = 1 \\ 0/8 = 0 \end{cases} \qquad P_4 = \begin{cases} 29/26 = 1.12 \\ 10/17 = .50 \\ 22/12 = 1.83 \\ 23/14 = 1.64 \\ 17/20 = .85 \\ 13/26 = .50 \end{cases}$$

$$P_2 = \begin{cases} 14/11 = 1.27 \\ 2/9 = .22 \\ 17/7 = 2.43 \\ 11/2 = 5.5 \\ 7/10 = .70 \\ 2/15 = .13 \end{cases} \qquad P_5 = \begin{cases} 32/32 = 1 \\ 14/18 = .78 \\ 23/20 = 1.15 \\ 25/19 = 1.32 \\ 26/26 = 1 \\ 20/28 = .71 \end{cases}$$

$$P_3 = \begin{cases} 22/18 = 1.22 \\ 6/13 = .46 \\ 18/11 = 1.64 \\ 19/6 = 3.17 \\ 10/17 = .59 \\ 10/18 = .56 \end{cases}$$

The rankings for each problem are

P_1: $(6\ 2\ \underline{5\ 1}\ 3\ 4)$

P_2: $(6\ 2\ 5\ 1\ 3\ 4)$

P_3: $(2\ 6\ 5\ 1\ 3\ 4)$

P_4: $(6\ 2\ 5\ 1\ 4\ 3)$

P_5: $(6\ 2\ \underline{5\ 1}\ 3\ 4)$

A graphical depiction of each unique sequence is shown in figure 58. Clearly, the optimal ordering is (2 6 5 1 3 4), yielding a makespan of 50.

The logic behind the Campbell-Dudek-Smith algorithm can be explained by examining certain features of the two-machine sub-problems. The first two-machine sub-problem is fundamentally important because it contains only the most critical

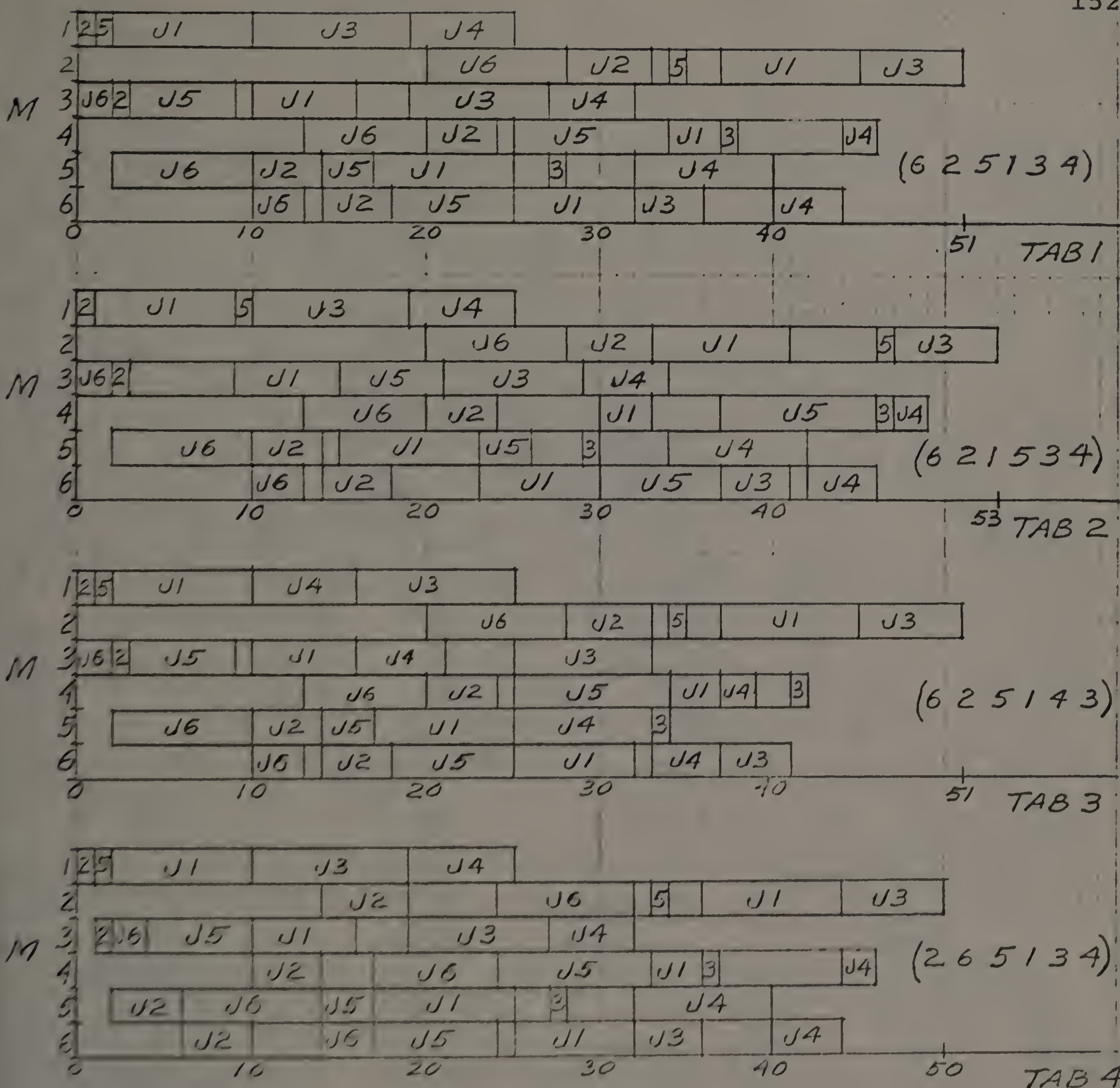


FIG. 58

elements of the original problem, namely, the first and last operations. (Recall that the slope method recognizes the importance of these operations by weighting them most heavily.) As a first approximation, the Campbell-Dudek-Smith procedure suggests that such a sub-problem, composed of only the most critical elements, represents a reasonably satisfactory abstraction of the original problem. Accordingly, a "good" solution to the original problem might be obtained by solving the first two-machine sub-problem by means of the Johnson method.

Whereas it is intuitive that a "good" solution is obtained following the above procedure, it is also intuitive that a better one might be generated by also considering those operations which <u>adjoin</u> the most critical, namely, the second and next-to-last. The second two-machine sub-problem acknowledges this possibility by incorporating these secondarily important operations into the analysis.

In each successive two-machine sub-problem, operation-pairs of progressively lesser importance are introduced. The final two-machine sub-problems integrate all operations of the original problem within the analysis.

Whereas the Palmer slope method uses explicit weighting factors to express the relative importance of operations, the Campbell-Dudek-Smith method achieves the same effect by relying on an interesting property of fractions. Recall that the Johnson method (modified) suggests computation of the ratios of first and last operations lengths in order to determine job processing sequence. Following the Johnson-Campbell-Dudek method, analysis of the m-machine problem begins with the computation of such ratios for the first two-machine sub-problem.

Heaviest "weighting" of the first and last operations is achieved by virtue of the fact that the first ratio set forms the basis for all successive sub-problems. As the original ratios are modified by successive additions to numerator and denominator, the effect on the ratio values becomes progressively smaller. To put it another way, as newly-considered operations become less and less critical (i.e. nearer the center of the "P" matrix), the capacity of such operations to cause any significant changes in the original ratios is progressively reduced. Hence, the job sequence established by the solution of the first two-machine sub-problem tends, in most instances, to undergo only minor modification.

To illustrate these important points, consider first the processing-time matrix of the first two-machine sub-problem

$$P_1 = \begin{bmatrix} 8 & 8 \\ 1 & 5 \\ 9 & 6 \\ 6 & 0 \\ 1 & 1 \\ 0 & 8 \end{bmatrix}$$

Figure 59

for which the ratios are computed as

$$P_1 = \begin{cases} 8/8 = 1 \\ 1/5 = .2 \\ 9/6 = 1.5 \\ 6/0 = \infty \\ 1/1 = 1 \\ 0/8 = 0 \end{cases}$$

The first job sequence is thus (6 2 5 1 3 4).

For the second two-machine sub-problem, the second and next-to-last operations are also included:

$$P_2 = \begin{bmatrix} 8+6 & 3+8 \\ 1+1 & 4+5 \\ 9+8 & 1+6 \\ 6+5 & 2+0 \\ 1+6 & 9+1 \\ 0+2 & 7+8 \end{bmatrix}$$

Figure 60

and the original ratios are thus modified to

$$P_2 = \begin{cases} \frac{8+6}{8+3} = 1.27 \\ \frac{1+1}{5+4} = .22 \\ \frac{9+8}{6+1} = 2.43 \\ \frac{6+5}{0+2} = 5.5 \\ \frac{1+6}{1+9} = .70 \\ \frac{0+2}{8+7} = .13 \end{cases}$$

Note that the inclusion of the second and next-to-last operations does not change the values of the ratios very much, and, most important, the rankings are not altered at all.

With each successive addition of operation-pairs, the numerator and denominator of each ratio grow by an amount equal to the length of the newly-included operations. Thus, the effect of each new inclusion on the previous ratio becomes progressively smaller, according to the principle that the ratio $\frac{A+B}{C+D}$ approaches $\frac{A}{C}$ as $\frac{B}{A}$ and $\frac{D}{C}$ approach zero. (As an extreme example, consider the difference between, say, 1/2 and $\frac{1+1}{2+9}$ versus, say, $\frac{1000}{2000}$ and $\frac{1000+1}{2000+9}$. In the first case, the base ratio is altered considerably by the addition of values to numerator and denominator; in the second case, however, the alteration is essentially insignificant.)

Evidently, there are cases in which the addition of operation-pairs results in some modification of job rankings. This generally occurs when added operations are grossly dissimilar in length, as in the case of the transition between P_2 and P_3. Clearly, however, the effect of such dissimilarity tends to be mitigated to the extent that operations fall near the center of P.

In conclusion, the Campbell-Dudek-Smith method produces schedules of satisfactory efficiency. An empirical comparison indicates marginal superiority over the Palmer slope method;[4] however, this advantage appears to be balanced by the disadvantage of slightly less computational ease. (The Campbell-Dudek-Smith method requires direct evaluation of several schedules.)

Conclusion

This chapter of the dissertation has outlined two methods for obtaining near-optimal solutions to the m-machine, n-job flowshop problem. Other methods exist, but, in general, they do not appear to be sufficiently unique to warrant inclusion. Also, many of them cannot be applied to problems of any reasonable size without recourse to computing devices.

It should be noted that the approach suggested for application to the general job-shop case (see prior chapters) can be applied equally well to the flowshop problem. We have chosen to recommend more direct methods, however, because such methods promise equally good or better results with far less computational tedium. The flowshop problem is clearly far less complicated than the job shop case; it therefore makes good sense to attack it in a manner befitting its simplicity.

In this dissertation, we have chosen to devote attention to only one of the many special cases of the job-shop problem. Other restrictive situations, say, $m \leq 3$ and/or $n \leq 3$, were not considered simply because such cases are rarely encountered in the real world. A wide variety of analytic approaches has been developed to handle all of the most obvious abstractions of the mxn job shop case, but, for the most part, we believe these developments to be of little or no general interest.

Footnotes

1. D.S. Palmer, "Sequencing Jobs through a Multi-Stage Process in the Minimum Total Time - A Quick Method of Obtaining a Near Optimum," Operational Research Quarterly, 16 (March, 1965), 101-107.

2. H.G. Campbell, et. al., "A Heuristic Algorithm for the n Job, m Machine Sequencing Problem," Management Science, 16 (June, 1970), B630-B637.

3. Johnson, "Optimal Two and Three Stage Production Schedules with Set-Up Times Included," Naval Research Logistics Quarterly, 1 (March, 1954), 61-68.

4. Palmer, "Sequencing Jobs through a Multi-Stage Process."

CHAPTER VIII

SUMMARY AND CONCLUSIONS

Summary

The purpose of this dissertation was to describe and justify a heuristic scheduling methodology that could easily be applied in situations where extensive computational facilities were unavailable. Chapter one introduced the reader to the problems inherent in analyzing the scheduling problem, and cited numerous studies in which the scheduling problem was approached in several different ways. Chapter two presented some fundamental concepts and definitions that were deemed necessary as a basis for understanding the problem and its attendant specialized features and variations. In Chapter three, some attention was devoted to dispatching and the nature of a queue, but the major focus was on the identification and understanding of basic priority rules, as well as on their application in a typical problem setting. Chapter four was an extension of Chapter three in which rules were developed, not merely with the purpose of generating "compact" schedules, but with the intention of accommodating specific due-date restrictions as well. In Chapter five, the focus was on heuristic rules ("rules of thumb") that might be used to supplement, and thereby enhance the effectiveness of, the previously developed priority rules. Chapter six dealt with the dynamic job shop case, a logical extension of the static case in which jobs could be added to or deleted from the original job set at any point in time. And, finally, the purpose of chapter seven was to describe and illustrate some highly specialized techniques that could be used to expeditiously handle a commonly-encountered variation of the job-shop problem - the flowshop problem.

Conclusion

The heuristic argument revisited. The heuristic scheduling methods described in this paper constitute a repertory from which the scheduler may draw when he sets out to determine a schedule. Knowledge of the strengths and weaknesses of each method, combined with the scheduler's understanding of the nature of the specific problem being analyzed, should permit an expeditious selection of priority rules and supplementary heuristics, the end result being a sub-optimal schedule of very good quality. Thus, the scheduler may successfully use this dissertation as a guide to effective scheduling in the small job shop, or any other shop that, for one reason or another, is constrained to rely on manual scheduling means.

The lingering objection of the purist will evidently focus on the lack of precision inherent in both the methodology and the results. The philosophical argument of March and Simon (that sub-optimizing or "satisficing" is the only suitable approach in most real-world problem-solving situations) may not be sufficiently compelling and may therefore require some statistical reinforcement.

Hare[1] and others, recognizing the futility of a "brute-force" search for optimality when very large numbers of feasible schedules are involved, have attempted to justify the use of sampling (inspection of only a small fraction of the total number) on the basis of a non-parametric statistical argument. Specifically, it can be shown that (a) a relatively small sample, chosen at random, has an astonishingly high probability of containing an optimal schedule, and (b) that this probability is only indirectly related to the size of the entire "population" of feasible schedules.

Let us begin with the assumption that the sample is chosen completely at random, that is, that no attempt has been made to enhance the qualities of the selected schedules through application of priority rules and/or supplementary heuristics. Under these conditions, it can be shown that the probability of picking at least one number of some specified "top" fraction p is $1-(1-P)^n$, where n is the size of the sample.

To illustrate, suppose that ten schedules are generated at random (n=10) and that it is desired to ascertain the probability that this sample will contain one or more schedules of the best 10% (P=.10). We calculate:

$$P = 1-(1-.10)^{10}$$
$$= 1-(.9)^{10}$$
$$= 1-.3486 = .6514$$

In other words, there is approximately a 65% probability that the sample of ten schedules will contain one or more schedules of the best 10%.

Suppose the sample size is increased to 20. The calculation is them:

$$P = 1-(1-.10)^{20}$$
$$= 1-(.9)^{20}$$
$$= 1-(.9)^{10}(.9)^{10}$$
$$= 1-(.3486)(.3486)$$
$$= 1-.1215 = .8785$$

By slightly increasing the sample size (i.e. from 10 to 20) it is possible to get greatly improved results. Since the sample size is quite small to begin with, such increase is normally quite workable.

As an alternative to increasing sample size, a similar improvement in probability of success might be obtained by weakening the restrictions on the "top" fraction. As an example, one might desire the probability of picking at least one member of the "top" 20%, instead of 10%, for a given sample size. If the sample size were restored to 10, the computation would be:

$$P = 1-(1-.20)^{10}$$
$$= 1-(.8)^{10}$$
$$= 1-.1074$$
$$= .8926$$

The results obtained here ($P \simeq 89\%$) are quite similar to those which occurred when the sample size was doubled. ($P \simeq 89\%$) It should be noted, however, that the benefits in the present case are somewhat illusory, since an improvement in the probability of success was secured only by permitting the "top" fraction to encompass more schedules. Thus, while the probability of encountering a "good" schedule is increased, the word "good" is now defined with considerably greater tolerance.

The foregoing discussion clearly illustrates a vital point, namely, that a relatively small sample, selected at random, has a rather high likelihood of yielding at least one good schedule. Thus, the implied technique would be to generate, say, ten or twelve schedules at random, evaluate them, and choose the one adjudged "best" on the basis of some criterion.

Assuming schedules are picked completely at random, there is approximately a 65% chance that a sample of 10 will contain at least one schedule of the top 10%. Suppose now, however, that schedules are <u>not</u> picked totally at random, that is, that they are instead carefully constructed (using priority rules and supplementary heuristics) in a manner that is likely to enhance certain important characteristics (such as makespan or idle time). Under these circumstances, what improvement in the probability of success might be expected?

It would be somewhat difficult to determine an exact quantitative answer to this question, since we do not know any precise statistics relating to the reliability of loading rules. Nevertheless, we can, with a fair amount of certainty, state that dispatching according to well-conceived rules (such as the ones discussed at length earlier in this paper) generally produces results that are superior to those obtained under conditions of random dispatching.

To summarize, we might use a heuristic argument as follows: A random selection process produced results that are reasonably good, sufficiently good, in fact, that we might, under certain conditions, be willing to accept the schedules thus produced as being at least marginally satisfactory. On the other hand, the use of carefully conceived heuristics is likely to produce even better results, thus enhancing the attractiveness of accepting such rules as part of a viable, "satisficing" approach to scheduling.

Computer programming of rules. Throughout this paper, we have proceeded under the assumption that electronic computational facilities were not available to the scheduler. Under such conditions, it would have been presumptuous to propose as "workable" any method which required the drawing-up of more than fifteen or twenty schedules (perhaps one for each priority rule-heuristic combination). It should, however, be carefully noted that there is nothing about the suggested method which explicitly prohibits computerization, and, consequently, the efficient generation of vastly larger numbers of schedules.

In the event that it becomes practicable to generate large numbers of schedules (i.e. in the event that computers are available), the argument in favor of the proposed approach takes on dramatic new significance. Even assuming random sampling, an increase in the sample size to 100 (an entirely workable figure) results in the following P:

$$
\begin{aligned}
P &= 1-(1-.10)^{100} \\
&= 1-(.9)^{100} \\
&= 1-[(.9)^{20}]^{5} \\
&= 1-[.1215]^{5} \\
&= 1-.0000264 \\
&= .9999
\end{aligned}
$$

Thus, the probability that a sample of size 100 will contain one or more schedules of the top 10% is now very close to 100%.

To get even greater accuracy, the top fraction might be reduced to, say, 1%, and the sample size increased to, say, 1000 (still a reasonable figure for computer enumeration). The resulting P would then be:

$$P = 1-(1-.01)^{1000}$$
$$= 1-(.99)^{1000}$$
$$= 1-(.3665)^{10}$$
$$= 1-.0000000433$$
$$= .9999$$

Now, the probability that a sample of size 1000 will contain one or more schedules of the top 1% is approximately 100%.

This example clearly indicates why sampling, using high-speed digital computers, has been frequently suggested as a means for solving the scheduling problem. Implicit also is the suggestion that priority rules and heuristics might be programmed to yield equally precise results, though without the need to generate and inspect nearly as many schedules.

Mechanical scheduling devices. The vehicle for communication of most academic thought is, sometimes unfortunately, paper and ink. The illustration and discussion of Gantt Chart manipulation has, in this dissertation for example, been seriously hampered by the media. In practice, the scheduler evidently need not draw up numerous successive stages in the generation of a final schedule, as was done throughout this paper.

The scheduling process is greatly facilitated through the use of magnetic boards or panels, on which blocks, bearing either job or machine designations, are positioned. Each block represents one unit of time, and single blocks or groups of blocks may easily be shifted to create an infinite number of different schedules. Other formats include boards and blocks with Velcro fasteners or adhesive, but the principles of operation are generally uniform.

A typical board-and-block device is depicted below:

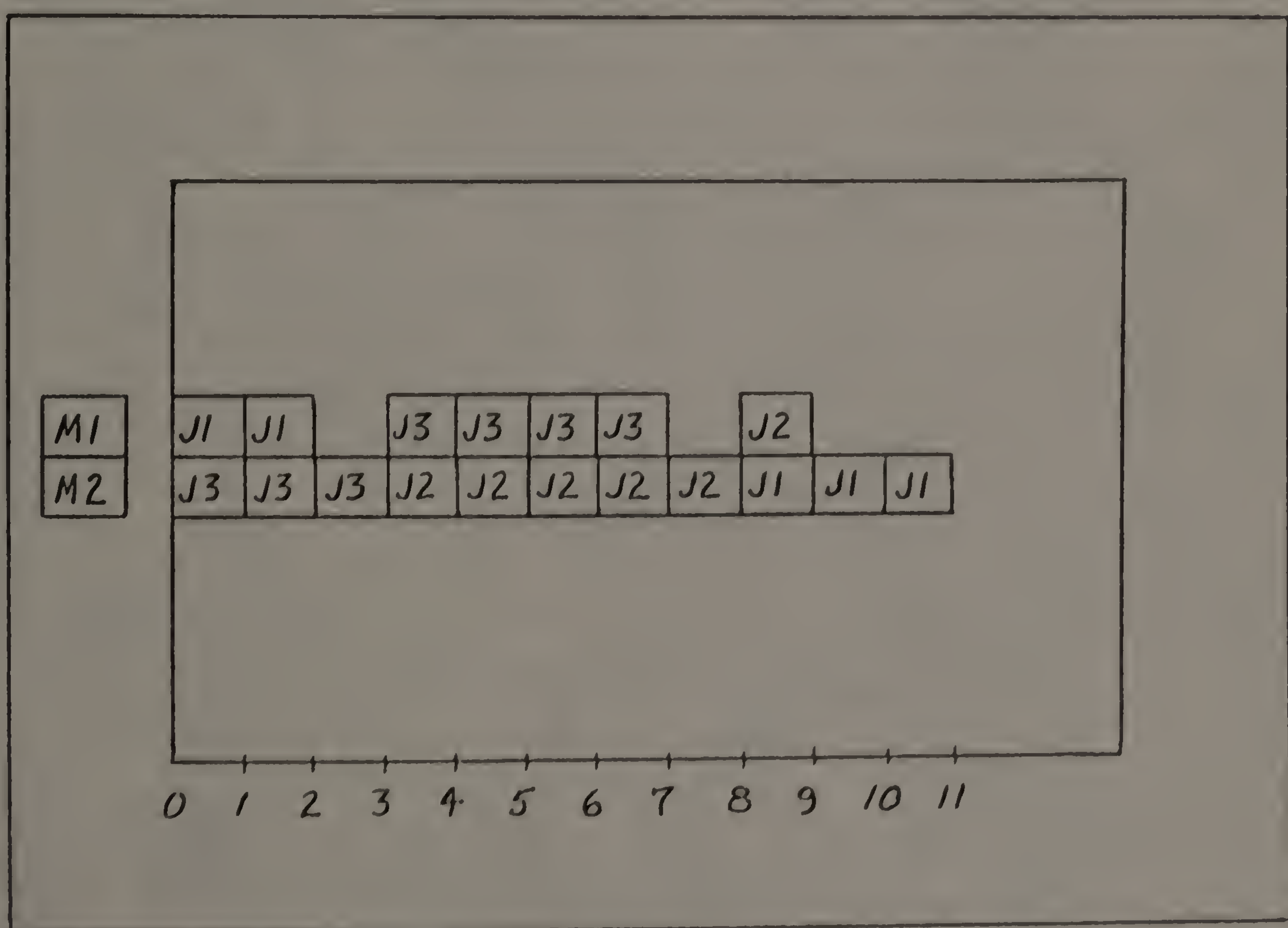


Figure 61

<u>Comparison of heuristically generated schedules with known optima</u>. Appendix A contains the heuristic solution of a scheduling problem with a known optimum.

REFERENCES

Akers, S.B. and Friedman, J. "A Non-Numerical Approach to Production Scheduling Problems." Operations Research, 3 (1955); 429-442.

Ashour, S. and Parker, R.G. "Precedence Graph Algorithm for the Shop Scheduling Problem." Operations Research Quarterly, 22 (June 1971), 165-175.

Ashour, Said. Sequencing Theory. Berlin, New York: Springer-Verlag, 1972.

Baker, C.T. and Dzielinski, B.P. "Simulation of A simplified Job Shop." Management Science, 6 (1960), 311-323.

Bakshi, M.S. and Arora, S.R. "The Sequencing Problem." Management Science, 16 (Dec. 1969), B247-B263.

Bellman, Richard. "Mathematical Aspects of Scheduling Theory." Journal of the Society of Industrial and Applied Mathematics, 4 (1956), 168-205.

Bellman, Richard. "Dynamic Programming Treatment of the Traveling Salesman Problem." Journal of the Association for Computing Machinery, 9 (Jan. 1962), 61-63.

Bomberger, E.E. "A Dynamic Programming Approach to a Lot Size Scheduling Problem." Management Science, 12 (July 1966), 778-784.

Bowman, E.H. "The Scheduling-Sequencing Problem." Operations Research, 7 (Sept. 1959), 621-624.

Bravoco, Ralph. "The Scheduling (Sequencing) Problem - Review and Extensions." Unpublished Report, University of Massachusetts, 1971.

Brown, A.P.G. and Lomnicki, Z.A. "Some Applications of the Branch-and-Bound Algorithm to the Machine Scheduling Problem." Operations Research Quarterly, 17 (June 1966), 173-186.

Bulkin, M.H. et al. "Load Forecasting, Priority Sequencing, and Simulation in a Job Shop Control System." Management Science, 13 (Oct. 1966), B29-B51.

Campbell, H.G. et al. "Heuristic Algorithm for the n-Job m-Machine Sequencing Problem." Management Science, 16 (June 1970), B630-B637.

Charlton, J.M. and Death, C.C. "Generalized Machine-Scheduling Algorithm." Operational Research Quarterly, 21 (March 1970), 127-34.

Conway, R.W., Maxwell, W.L. and Miller, L.W. Theory of Scheduling. Reading, Mass.: Addison-Wesley, 1967.

Crabill, T.B. "A Lower-Bound Approach to the Scheduling Problem." Research Report, Department of Industrial Engineering, Cornell University, 1964.

Dantzig, G. "A Machine-Job Scheduling Problem." Management Science, 6 (Jan. 1960), 191-196.

Dantzig, G., Fulkerson, D.R. and Johnson, S.M. "On a Linear Programming Combinatorial Approach to the Traveling Salesman Problem." Operations Research, 7 (Jan. 1959), 58-66.

Dudek, R.A. and Teuton, O.F. "Development of M-Stage Decision Rule for Scheduling n Jobs through M Machines." Operations Research, 12 (May 1964), 471-497.

Elmaghraby, S.E. "A Loading Problem in Process Type Production." Operations Research, 16 (Sept. 1968), 902-914.

Emmons, H. "One-Machine Sequencing to Minimize Certain Functions of Job Tardiness." Operations Research, 17 (July 1969), 701-715.

Fischer, H. and Thompson, G.L. "Probabalistic Learning Combinations of Local Job Shop Scheduling Rules. In Muth, J.F. and Thompson, G.L. (eds.) Industrial Scheduling. Englewood Cliffs, N.J.: Prentice-Hall, 1963.

Florian, M., Trepant, P. and McMahon, G. "An Implicit Enumeration Algorithm for the Machine Sequencing Problem." Management Science, 17 (Aug. 1971), B782-B792.

Gapp, W., Minkekar, P.S. and Mitten, L.G. "Sequencing Operations to Minimize In-Process Inventory Costs." Management Science, 11 (Jan. 1965), 476-484.

Favett, W.J. "Three Heuristic Rules for Sequencing Jobs to a Single Production Facility." Management Science, 11 (June 1965), B166-B176.

Gere, W.S. "Heuristics in Job Shop Scheduling." Management Science, 13 (Nov. 1966), 167-190.

Giffler, B. and Thompson, G.L. "Algorithms for Solving Production Scheduling Problems." Operations Research, 8 (July 1960), 487-503.

Giffler, B., Thompson, G.L. and Van Ness, V. "Numerical Experience with the Linear and Monte Carlo Algorithms for Sovling Production Scheduling Problems." In Muth, J.F., and Thompson, G.L. (eds.) Industrial Scheduling. Englewood Cliffs, N.J.: Prentice-Hall, 1963.

Gilmore, P. and Gomory, R.E. "Sequencing of One State-Variable Machine." Operations Research, 12 (Sept. 1964), 655-679.

Glassey, C.R. "Cynamic Linear Programs for Production Scheduling." Operations Research, 19 (Jan. 1971), 45-56.

Gordon, G. "A General Purpose Systems Simulator." IBM Systems Journal, September 1962.

Greenberg, H.H. "Branch-Bound Solution to the General Scheduling Problem." Operations Research, 16 (March 1968), 351-61.

Gupta, J.N.D. "Functional Heuristic Algorithm for the Flow Shop Scheduling Problem." Operations Research Quarterly, 22 (1971), 39-47.

Gupta, J.N.D. "Improved Combinatorial Algorithm for the Flow Shop Scheduling Problem." Operations Research, 19 (Nov. 1971), 1753-8.

Hardgrave, W.W. and Nemhauser, G.L. "A Geometric Model and a Graphical Algorithm for a Sequencing Problem." Operations Research, 11 (Nov. 1963), 889-900.

Hare, Van Court, Jr. Systems Analysis: A Diagnostic Approach. New York: Harcourt, Brace and World, 1967.

Held, M. and Karp, R.M. "A Dynamic Approach to the Sequencing Problem." Journal of the Society of Industrial and Applied Mathematics, 10 (1962), 48-53.

Held, M. et al. "Assembly-line Balancing: Dynamic Programming with Precedence Constraints." Operations Research, 11 (May 1963), 442-59.

Heller, J. "Some Problems in Linear Graph Theory that Arise in the Analysis of the Sequence of Jobs Through Machines." AEC Research and Development Report NYO-987, 1960.

Heller, J. and Logemann, G. "An Algorithm for the Construction and Evaluation of Feasible Schedules." Management Science, 8 (Jan. 1962), 168-183.

Ignall, E. and Schrage, L. "Application of the Branch-and-Bound Technique to Some Flow Shop Scheduling Problems." Operations Research, 13 (May 1965), 400-412.

Jackson, J.R. "Scheduling a Production Line to Minimize Maximum Tardiness." Management Science Research Project, Report No. 43, University of California, 1955.

Jackson, J.R. "An Extension of Johnson's Results of Job Lot Scheduling." Naval Research Logistics Quarterly, 3 (Sept. 1956), 201-203.

Jackson, J.R. "Simulation Research on Job Shop Production." Naval Research Logistics Quarterly, 4 (Dec. 1957),

Johnson, S.M. "Optimal Two and Three-Stage Production Schedules with Setup Times Included." Naval Research Logistics Quarterly, 1 (March 1954), 61-68.

Johnson, S.M. "Discussion: Sequencing n Jobs on Two Machines with Arbitrary Time Lags." Management Science, 5 (1959).

Karg, R.L. and Thompson, G.L. "A Heuristic Approach to Solving Traveling Salesman Problems." Management Science, 10 (Jan. 1964), 225-248.

Lawler, E.L. and Moore, J.M. "A Functional Equation and Its Application to Resource Allocation and Sequencing Problems." Management Science, 16 (Sept. 1969), 77-84.

LeGrande, Earl. "The Development of a Factory Simulation System Using Actual Operating Data." Management Technology, 3 (May 1963), 1-19.

Little, J.D.C., Murty, K.G., Sweeney, D.W. and Karel, C. "An Algorithm for the Traveling Salesman Problem." Operations Research, 11 (Nov. 1963), 972-989.

Lomnicki, Z.A. "A Branch-and Bound Algorithm for the Exact Solution of the Three-Machine Scheduling Problem." Operations Research Quarterly, 16 (March 1965), 89-100.

Manne, A.S. "On the Job Shop Scheduling Problem." Operations Research, 8 (March 1960), 219-223.

March, J.G. and Simon, H.A. Organizations. New York: Wiley, 1964.

Maxwell, W.L. "On Sequencing n Jobs on One Machine to Minimize the Number of Late Jobs." Management Science, 16 (Jan. 1970), 295-297.

McMahon, G.B. and Burton, P.G. "Flow Shop Scheduling with the Branch-and-Bound Method." Operations Research, 15 (May 1967), 473-481.

McNaughton, R. "Scheduling with Deadlines and Loss Functions." Management Science, 6 (1959), 1-12.

Mellor, P. "A Review of Job-Shop Scheduling." Operations Research Quarterly, 17 (1966), 161-167.

Mitten, L.G. "Sequencing n Jobs on Two Machines with Arbitrary Time Lags." Management Science, 5 (1959), 293-298.

Moore, J.M. "An n Job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs." Management Science, 15 (Sept. 1968), 102-109.

Newell, A., Shaw, J.C. and Simon, H.A. "The Process of Creative Thinking." RAND Corporation Paper P-1320, 1958.

Palmer, D.S. "Sequencing Jobs through a Multi-Stage Process in the Minimum Total Time: Quick Method of Obtaining a Near Optimum." Operational Research Quarterly, 16 (March 1965), 101-107.

Presby, J.T. and Wolfson, M.L. "Algorithm for Solving Job Sequencing Problems." Management Science, 13 (April 1967), B454-B464.

Root, J.G. "Scheduling with Deadlines and Loss Functions on K Parallel Machines." Management Science, 11 (Jan. 1965), 460-475.

Rowe, Alan J. "Toward a Theory of Scheduling." Journal of Industrial Engineering, 11 (March, 1960), 125-136.

Schwartz, E.S. "An Automated Sequencing Procedure with Application to Parallel Programming." Journal of the Association for Computing Machinery, 8 (1961), 513-37.

Schwartz, E.S. "A Heuristic Procedure for Parallel Sequencing with Choice of Machines." Management Science, 10 (July 1964), 767-777.

Simon, H.A. and Newell, A. "Heuristic Problem-Solving: The Next Advance in Operations Research." Operations Research, 6 (1958), 1-10.

Simon, H.A. and Newell, A. "Computer Simulation of Human Thinking and Problem Solving." Computers and Automation, 10 (1961), 18-26.

Sisson, R.L. "Methods of Sequencing in Job-Shops - A Review." Operations Research, 7 (Jan. 1959), 10-29.

Smith. R.D. and Dudek, R.A. "A General Algorithm for Solution of the n Job, M Machine Sequencing Problem of the Flow Shop." Operations Research, 15 (Jan. 1967), 71-82.

Smith, W.E. "Various Optimizers for Single Stage Production." Naval Research Logistics Quarterly, 3 (1956), 59-66.

Szwarc, W. "Solution of the Akers-Friedman Scheduling Problem." Operations Research, 8 (Nov. 1960), 782-788.

Tonge, F.M. "Summary of a Heuristic Line-Balancing Procedure." Management Science, 7 (Oct. 1960), 21-42.

von Lanzenauer, C.H. "Production Scheduling Model by Bivalent Linear Programming." Management Science, 17 (Sept. 1970), 105-11.

Wagner, D.E. and Shetty, C.M. "Solution to a Production Scheduling Problem with Fixed Costs." Operations Research, (March 1962), 87-94.

Wagner, H.M. "An Integer Linear Programming Model for Machine Scheduling." Naval Research Logistics Quarterly, 6 (June 1959), 131-140.

Weist, J.D. "A Heuristic Model for Scheduling Large Projects with Limited Resources." Management Science, 13 (Feb. 1967), B359-B377.

APPENDIX A

The following example, consisting of three jobs and four machines, is taken from Thompson:[1]

$$F = \begin{bmatrix} 11 & 12 & 13 & 14 \\ 23 & 21 & 24 & 22 \\ 24 & 21 & -- & -- \end{bmatrix}$$

$$P = \begin{bmatrix} 5 & 2 & 8 & 7 \\ 8 & 4 & 5 & 3 \\ 6 & 7 & - & - \end{bmatrix}$$

Jobs 1 and 2 are processed by each of the four machines; job 3 is processed by machines 4 and 1 only. The known optimal makespan of the problem is 24.

Figure 62 illustrates application of the shortest imminent operation rule. Using this rule, without recourse to any secondary rule, the final schedule has a makespan of 30, indicating an error of approximately 20%. The reason for the error is not difficult to discern. At t = 5, M1 becomes idle, and J3M1 is ready to begin processing at t = 6. Since J2M1 is not available for processing until two time periods hence, J3M1 is scheduled. The decision is indeed optimal at this point, but unfortunately causes future complications. (J2M4 is delayed excessively.)

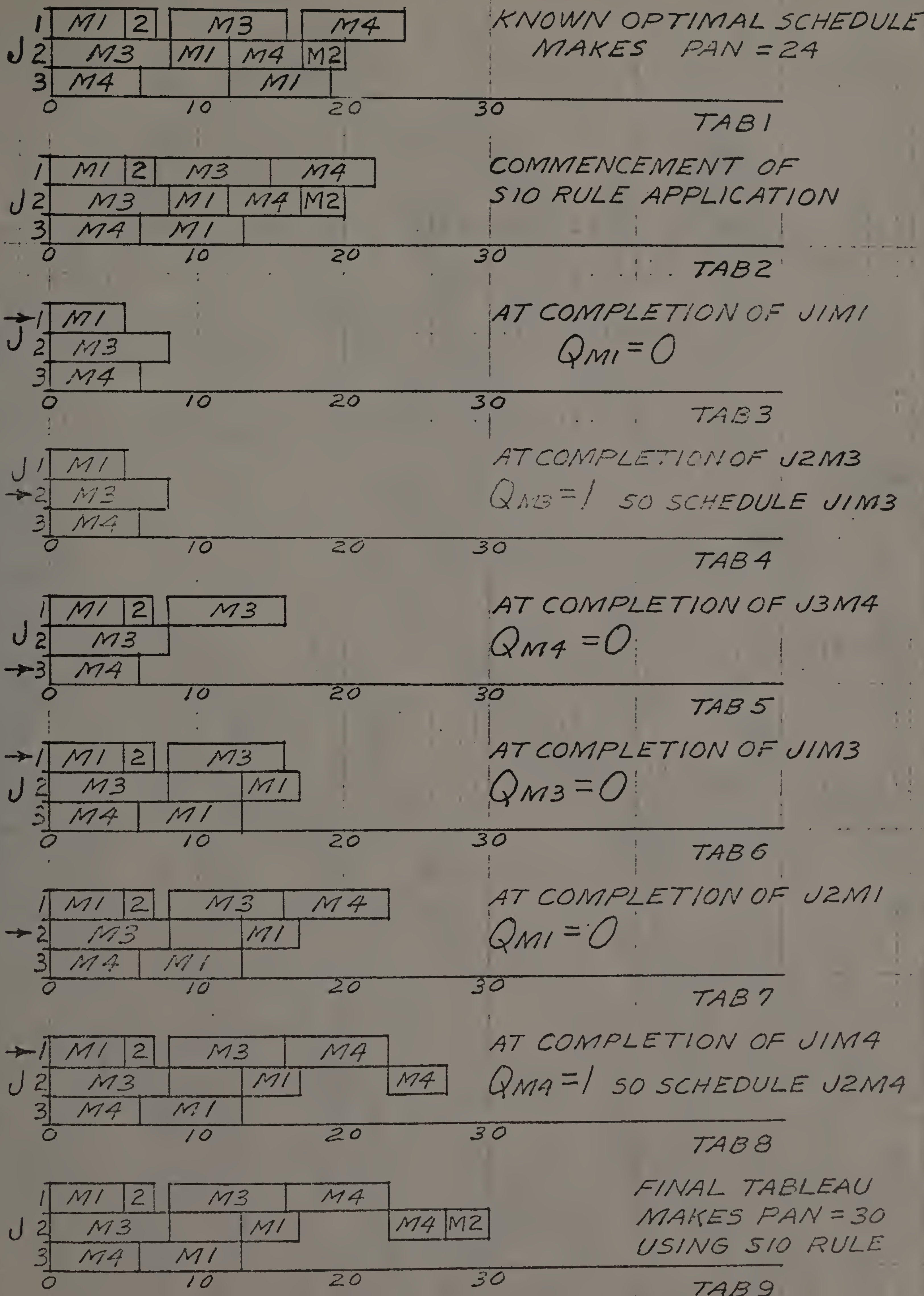

KNOWN OPTIMAL SCHEDULE
MAKES PAN = 24
TAB 1
COMMENCEMENT OF
S10 RULE APPLICATION
TAB 2
AT COMPLETION OF J1M1
$Q_{M1} = 0$
TAB 3
AT COMPLETION OF J2M3
$Q_{M3} = 1$ SO SCHEDULE J1M3
TAB 4
AT COMPLETION OF J3M4
$Q_{M4} = 0$
TAB 5
AT COMPLETION OF J1M3
$Q_{M3} = 0$
TAB 6
AT COMPLETION OF J2M1
$Q_{M1} = 0$
TAB 7
AT COMPLETION OF J1M4
$Q_{M4} = 1$ SO SCHEDULE J2M4
TAB 8
FINAL TABLEAU
MAKES PAN = 30
USING S10 RULE
TAB 9

FIG. 62

While a 20% error is not deemed excessive, the application of heuristics might have resulted in a better schedule. For example, the "alternative assignment" heuristic might have warned of difficulties at early stages, thus permitting the juxtaposition of J2M1 and J3M1. At this point, some back-tracking, coupled with the "idle time reduction" heuristic would be in order.

Admittedly, the problem is a simple one, but it illustrates the essential point: the methodology which has been suggested in this paper is capable of producing satisfactory solutions to problems involving job shop scheduling.

Footnote

[1]G.L. Thompson, "Recent Developments in the Job-Shop Scheduling Problem," in *Industrial Scheduling*, ed. by J.F. Muth and G.L. Thompson (Englewood Cliffs, New Jersey: Prentice-Hall, 1963).