recording density of 100 characters per inch, each tape would be on the order of 3,000 feet long.

Having demonstrated a set of conditions and some criteria under which magnetic tape files, fundamentally a sequential type memory, can be used for a so-called random access problem, the question might now be raised as to what kind of business data-processing problems require a large-scale random access (e.g., matrix type as opposed to sequential) memory, or for that matter even a very rapid access memory. To begin with, the several-per-second traffic rate chosen in the example is not greatly exceeded by peak rates of credit transactions of customers in a large department store or even by stock requisitions to Air Force inventory accounts at large depots. These two examples are isolated, of course, but they serve as convenient illustrative reference points. Secondly, business data-processing “response times” are more likely to be days or hours rather than minutes or seconds; probably seldom less than the 20 minutes chosen in the example. It is certainly possible to point to a few business operations where very short response times are useful (these examples generally seem to involve a customer, in person or on the phone, who must not be kept waiting, e.g., credit authorization for department store customers), but even in these cases it is not always necessary to refer to a voluminous file. If it could be economically justified, a large rapid access memory for sorting would of course be quite useful in many business applications. I suspect that the requirements of many other so-called random access problems in business actually can be met with magnetic tape equipment.

Data-Processor Requirements in Production and Inventory Control

H. T. LARSON AND A. VAZSONYI†

INTRODUCTION

AUTOMATIC DATA processing, as performed by punch card machines, has been an important tool in the business world for many years. In the field of accounting and statistical data collection, it would probably be impossible to conduct business effectively, without the use of such machines. During the last few years it has been increasingly recognized that new electronic computers, through their superior performance, will surpass in efficiency the current punch card machines. A number of firms are in the process of introducing electronic computers in their business operations, and an even larger number of organizations are making plans for the application of large-scale electronic computers.

Automatic data processing in production and inventory control, on the other hand, is something of a new development. There are a number of firms using punch card machines in production and inventory control, but it appears that there is no corporation that has as yet a fully integrated automatic production and inventory control system. There are a number of reasons why progress has been relatively slow in this field. First of all, problems in production control are a great deal more complicated than, say, problems of conventional accounting. Furthermore, the methods of production and inventory control have been only recently developed, systems and procedures have not sufficiently settled down, and therefore there is a continuous need for the introduction of new techniques. Another point is that the techniques of production control are not yet sufficiently articulated; and, consequently, procedures often are carried through on an intuitive basis. This makes the introduction of automatic data processing exceedingly difficult.

To appreciate the complexities of production and inventory control, compare, for example, the problem of preparing a payroll with the problem of parts listing. True, there is a great deal of data processing in the preparation of a payroll; however, the problem is conceptually quite simple. A clerk in payroll accounting can be trained in a short time; the principal difficulty is the problem of handling the large amount of data accurately and speedily. Methods of avoiding and finding errors in this type of accounting system have been worked out for many years. Compare this situation with the problem of parts listing for a large manufacturing organization. Again, there is a great deal of data, but there are many conceptual complications. Complex assemblies are made up of other assemblies and of simple parts; improvements in engineering design continuously make obsolete some of the parts; inventory levels constantly change and adjustments must be made; sudden changes in master schedules must be made to correspond to new customer requirements. Many other similar features make data processing in production control exceedingly difficult. The development of checking procedures to assure the accuracy of the results obtained in

† Ramo-Wooldridge Corp., Los Angeles, Calif.
itself presents a problem much more complicated than the checking problems usually encountered in more conventional types of data processing.

It is not surprising then that during the last few years there has been an increasing emphasis in this field and it is gratifying that progress has been made at least on two fronts: (1) performance of electronic data processors has been evaluated by various organizations from the point of view of production and inventory control, and (2) new scientific methods, in particular mathematical models, have been developed to describe and analyze the processes of production control. The purpose of this paper is to report on the work of the authors in this field and to show through an illustrative example how some of the basic problems of production control can be handled with the aid of mathematical models and how electronic computers can be applied to these problems on the basis of such mathematical models.

Imagine a hypothetical factory producing some rather complex assemblies, similar to airplanes or radar sets.

![Fig. 1-Assembly parts list. This sheet refers to the assembly "panel" which is assigned the part number 435090012. This panel is made up of seven different articles which could be subassemblies or parts. The panel requires 3 bushings 420990309, and 1 panel blank 435090012-1, etc. Each assembly will have a similar sheet.](image1)

There might be thousands of these articles produced on assembly lines and in the machine shops. Imagine that master schedules for the shippable items are set well in advance, but are subject to periodic changes. The problem we propose to examine here is how to determine the number of parts and assemblies required to meet this shipping schedule and when these various parts should be manufactured. A further problem to be answered is the determination of machine and labor hours imposed on this factory by this master shipping schedule.

We develop answers to these problems through the development of a mathematical model. It will be seen that the mathematics involved are matrix multiplication and inversion. Fortunately, due to a special property of the matrices involved, the problem of inversion will appear as a problem in matrix multiplication. The principal part of the computational procedure being matrix multiplication, we spend a considerable part of our paper on this problem. This might lead to the conclusion that the need is for a special-purpose computer and not for a general-purpose data processor. This conclusion would be in error, as production control involves a great deal of conventional data processing such as sorting and collating. Furthermore, production control usually must be tied in with cost accounting and labor distribution, and it is unlikely that the computer would do parts listing and scheduling exclusively.

The exposition of the paper falls into two natural parts. The first is the development of the mathematical model. The second describes the computational technique involved, including estimates of the magnitude of the job to be performed for typical examples and also certain of the computer characteristics required.

**Development of the Mathematical Model**

The basic information required for production control is usually listed on so-called assembly parts lists. An example, shown on Fig. 1, refers to a Panel with part number 435090012. This "Makes Assembly" is made up of seven different articles as shown on the parts list under the heading N. A. QTY (Next Assemblies Quantity). It is shown how many of these articles are needed. Thus, the bushing part number 420990309 is required in a quantity of three for each of the panels 435090012. This sheet refers to the assembly "panel" which is assigned the part number 435090012. This panel

![Fig. 2—The Gozinto graph is a pictorial representation of the parts requirements. The next assembly quantities can be observed directly by counting the arrows on each connecting line. Total requirements cannot be observed directly but can be deduced.](image2)
Fig. 3—Next assembly quantity table. This is a concise mathematical representation of the information contained in the assembly parts lists.

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>

Fig. 4—Total requirement factor table. Observe, say, the second column relating to $A_2$. The third element from the top in this column relates to $A_3$ and displays the number 33. This means that 33 $A_3$'s are required (in total) for each $A_3$. This can be confirmed by a direct count from Fig. 10.

shows a highly simplified situation when there are only three top assemblies and a total of nine articles. The Next Assembly Quantities are shown by the arrows on the figure. The various Next Assembly Quantities shown on Fig. 2 are represented in a tabular form in Fig. 3, and this Next Assembly Quantity matrix is a concise mathematical representation of the information contained in the assembly parts list. For instance, it can be seen either from Fig. 2 or Fig. 3 that each Article 7 or Article 7 takes two $A_1$’s directly. We use the word “directly” advisedly as $A_1$ requires, in total, four of $A_4$, as $A_1$ requires two $A_4$’s directly and each $A_4$, in its turn, requires two of $A_4$ directly. We are, of course, interested to know how many of each article is required directly or indirectly for each other article. This information cannot be read directly from the Next Assembly Quantity table, and our problem is how to determine these total requirement factors from the Next Assembly Quantity table.

In the same way as we put the Next Assembly Quantities into tabular form, we put the Total Requirement Factors into a table. To illustrate the case, Fig. 4 shows the Total Requirement Factor table associated with the illustrative example in Fig. 2. We agree that each $A_i$

takes four of $A_i$’s in total. In Fig. 4, in the fifth row, under the seventh column, the number 4 is listed. The figure shows that 33 of $A_3$ is required for each $A_3$. Verification of this in Fig. 2 requires a careful tracing of the various arrows. This tracing in Fig. 2 can be described by the following statement:

Total number of $A_i$’s required for each $A_j =$

\[ (Number \ of \ A_i’s \ going \ directly \ into \ each \ A_j) \]

\[ \cdot \ (Total \ number \ of \ A_i’s \ required \ for \ each \ A_j) \]

\[ + \ (Number \ of \ A_i’s \ going \ directly \ into \ each \ A_j) \]

\[ \cdot \ (Total \ number \ of \ A_i’s \ required \ for \ each \ A_j) \]

\[ + \ (Number \ of \ A_i’s \ going \ directly \ into \ each \ A_j) \]

\[ \cdot \ (Total \ number \ of \ A_i’s \ required \ for \ each \ A_j). \]

Note again, carefully, the distinction between the statement “total number of $A_i$’s required...” and “number of $A_i$’s going directly into...”

We proceed now to put the above statement into mathematical form. We denote by $N_{i,j}$ the Next Assembly Quantity, which indicates the number of $A_i$’s required directly into an $A_j$. Furthermore, we denote by $T_{p,i}$ the Total Requirement Factor, that is, the total number of $A_j$’s required for each $A_j$. Then the statement above can be written as

\[ T_{p,i} = N_{s,p}N_{s,i} \]

or

\[ T_{p,i} = \sum_{p} N_{s,p}T_{p,j}. \]

A graphical representation of these two equations is given in Fig. 5.

Fig. 5—Schematic representation of the equation

\[ T_{p,i} = \sum_{p} N_{s,p}T_{p,j}. \]

The second number of the fifth row of the $T$-Table equals the "scalar multiple" of the fifth row of the $N$-Table and second column of the $T$-Table:

\[ 10 = 2 \times 3 + 0 \times 1 + 0 \times 33 + 0 \times 0 + 0 \times 10 + 0 \]

\[ \times 12 + 2 \times 1 + 1 \times 2 + 0 \times 0. \]

It is quite plausible now to conclude that the above formula can be generalized for any pair of articles $A_i$ and $A_j$. Therefore, we deduce that

\[ T_{i,j} = \sum_{p} N_{i,p}T_{p,j} \quad i \neq j. \]

We recall now our convention that

\[ N_{i,i} = 0 \]  

\[ T_{i,i} = 1 \]
These relations can be written in a more concise form by using matrix algebra:

\[ [T] = \left[ \frac{I}{I - N} \right], \quad (6) \]

where \([I]\) denotes the unit matrix. Our first problem, the determination of the Total Requirement Factors, leads to the problem of matrix inversion as shown by (6). Or, to put it in another way, the problem is to solve the system of (3) and (5) for the unknown \(T\)'s. If these matrices had no special property, it would be practically impossible to carry through this computational task when there are thousands of articles. Fortunately, most of the elements of the matrices are zeros and the matrix is “triangular”; consequently, special procedures can be developed. Before we describe the computational technique, we develop our mathematical model further.

As we said in the introduction, we need to know the total requirements for each part as related to a shipping schedule. Suppose we are shipping only articles \(A_2, A_4, A_9, \) and \(A_s\), the last being a spare. How do we compute the quantity of \(A_s\)'s required? Clearly

\[
\text{Quantity of } A_s \text{'s required} = \\
(\text{Total number of } A_s \text{'s required for each } A_2) \\
\times (\text{shipping requirements of } A_2) \\
+ (\text{Total number of } A_s \text{'s required for each } A_4) \\
\times (\text{shipping requirement of } A_4) \\
+ (\text{Total number of } A_s \text{'s required for each } A_9) \\
\times (\text{shipping requirement of } A_9) \\
+ (\text{Shipping requirement of } A_s). 
\]

In order to put this in mathematical form, we introduce the notation that the shipping requirements are given by \(S_1, S_2, \cdots\) and that the unknown requirements for the articles are \(X_1, X_2, \cdots\) With this notation the above verbal statement can be written as

\[
X_s = T_{s,2}S_2 + T_{s,4}S_4 + T_{s,9}S_9 + S_s. \quad (7)
\]

This equation can again be written as

\[
X_s = \sum_b T_{s,b}S_b. \quad (8)
\]

Again, it is plausible to generalize for any article \(A_i\)

\[
X_i = \sum_b T_{i,b}S_b. \quad (9)
\]

This last equation then gives a method of determining the quantity of each article required once the \(T\) matrix is computed and if the shipping schedule is given. It is important from the computational point of view to recognize that most of the \(S\)'s are zero, as usually only a small fraction of the articles manufactured are shipable.

**The Problem of Scheduling**

So far, we have concerned ourselves only with the problem of quantities required. Now we propose to introduce the time element. We will assume that our hypothetical factory operates in production periods, and we will assume that the shipping requirement is given for each article in each period. The question we must answer now is how many of each article is to be made in each period. We denote by \(s^m\) the number of \(A_i\)'s to be shipped in the \(m\)th period. Similarly, we denote the unknown quantity of \(A_i\)’s to be manufactured in the \(m\)th period by \(x^m\).

In every manufacturing process, articles must be made in a certain technological sequence. Looking at Fig. 2, one can see that, say, article \(A_1\) must be made before \(A_8\), as the former is a part of the latter. It is the usual practice in production control to make up so-called setback charts. Fig. 6 illustrates the setback chart for top assembly \(A_s\). Each article is allowed a cer-
tain make-time, and then a safety cushion is introduced. For instance, in Fig. 6, Article A₉ is allowed three production periods, and a cushion of two production periods is introduced for safety. It can be seen from the figure, for instance, that A₉ is to be started 19 periods earlier than A₉ is to be shipped. Suppose we know that s₉ is given indicating the number of A₉'s to be shipped in the 30th production period. Then we have

\[ x₉₄,₈^{11} = 3s₉^{10} \]  

where the left-hand side indicates the number of A₉'s to be made in the 11th production period, provided the shipping requirements of A₉ are the only requirements to be taken into consideration. The factor 3 appears as each A₉ takes three A₉'s. Eq. (10) easily generalizes into

\[ x₉ₙ₄₈^{n+11} = 3s₉^{n+10} \]  

or into

\[ x₉ₙ₄₈^{m} = Tₙ₉₄₈^{m+10} \]  

where σ₉₄ denotes the setback of Article A₉ "in" Article A₉.

In order to generalize this relationship to any article "in" any other article, we introduce the concept of the setback matrix σₖₖₖₖ, as illustrated in Fig. 7. With this notation we get a general formula for the number of Aₖₖ's to be made in production period m, with the proviso that these Aₖₖ's are earmarked to a shippable Aₖ:

\[ xₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖₖ₆
nonzero elements only in columns 2, 4, and 9. This condition would be satisfied for rows 7 and 8 of \([N]\). When a row of \([N]\) is found which satisfies this condition, hold it and multiply it successively with all the columns of \([T]\). As each element of \([T]\) is developed, insert it in the column of \([T]\). The multiplication of the 7th row of \([N]\) by the second column of \([T]\) is diagrammed in Fig. 9. Note that wherever an unknown appears in \([T]\), the corresponding element of \([N]\) is zero. This will always be true in this procedure. Add these new known j's to the list of known rows of \([T]\).

![Fig. 8—First steps in developing \([T]\).](image)

\[
\begin{bmatrix}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
1 & x & x & x & x & x & x & x & x \\
2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 1 & x & 1 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
5 & x & x & x & 1 & x & x & x & x \\
6 & x & x & x & x & x & x & x & x \\
7 & x & x & x & x & x & x & x & x \\
8 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
1 & x & x & x & x & x & x & x & x \\
2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
3 & 0 & 1 & x & 1 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
5 & x & x & x & 1 & x & x & x & x \\
6 & x & x & x & x & x & x & x & x \\
7 & x & x & x & x & x & x & x & x \\
8 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

6. As the elements of the rows of \([T]\) are developed and placed in the proper positions in the columns of \([T]\), save those elements of each row of \([T]\) which fall in columns \(k\) corresponding to shippable articles \(A_k\), and write these out, as a row, before proceeding. The reason for this is that we need the \(T_{i,k}\) stored in rows for the following computation, and since we are developing them in rows during the present task, we may as well write out the rows and save some of the sorting that would be entailed in changing \([T]\) from column form to row form.

7. With the pass described in steps 5 and 6 complete, repeat the search described in step 5, now matching nonzero elements of the rows of \([N]\) against the augmented list of known rows of \([T]\). This will locate the next lower echelon of articles on the Gozinto graph, make it possible to compute additional rows of \([T]\), etc. This iteration is continued until all the elements of \([T]\) are computed.

8. Sort the rows of \([T]\), written out in step 6, into ascending sequence on the row number \(i\).

**Computation of the Production Requirements Schedule, \(X_{i,m}\)**

Eq. (13) shows that the ingredients for this computation are the Total Requirements Factor matrix \([T]\), the setback matrix \([\sigma]\), and the shipping schedules \(s_{k}^{m}\). The procedure is as follows:

1. Bring in the first row of \([T]\), and find the first nonzero element in that row; the co-ordinate \(k\) represents an article \(A_k\) into which article \(A_l\) goes. In the example given in Fig. 10, the first \(A_k\) is \(A_5\).

![Fig. 10—The \(T\)-matrix and \(\sigma\) matrix, including columns \(k\) only for shippable articles \(A_k\).](image)

2. Bring in the first row of \([\sigma]\). This will have nonzero elements in the same positions as in \([T]\), since wherever an article \(A_k\) goes into an article \(A_l\) there must be a setback stated. Look up the setback \(\sigma_{i,k}\) corresponding to the element found in step 1. In our example, this is \(\sigma_{1,2} = 3\).

3. Bring in the shipping schedule \(s_{k}^{m}\) associated with the article \(A_k\) found in step 1. If the first production period for which we wish to compute the schedule on \(A_l\) is period \(m_1\), form the sum \((m_1 + \sigma_{i,k})\). This is the period of the shipping schedule in which we are interested. Pull the quantity \(s_{k}^{m_1+\sigma_{i,k}}\) found in this period, and multiply it by \(T_{i,k}\). In our example, we compute period 16 first, which means we look up \(s_{5}^{16}\) = 1 (Fig. 11), and form the product \(T_{1,5}^{16} = 3 \times 1 = 3\).

![Fig. 11—Shipping schedules for articles \(A_k\).](image)

4. Find the next nonzero element in the first row of \([T]\). The example shows this as \(T_{1,4} = 3\).

5. Look up the corresponding setback in the first row of \([\sigma]\). In the example this is \(\sigma_{1,4} = 3\).
6. Bring in the shipping schedule for the new \( A_k \) found in step 4. Here, we would bring in schedule \( x_{r}^m \).

In period \((16 + 3)\) we find \( s_{t}^{19} = 4\). Form the product

\[
T_{1,4}s_{19}^{19} = 3 \times 4 = 12,
\]

and add this to the product developed in step 3.

7. Continue this iteration until all of the nonzero elements of the first row of \([T]\) have been considered. This will form the first element of \( x_{r}^m \). In the example, this forms

\[
x_{1}^{16} = T_{1,5}s_{19}^{19} + T_{1,6}s_{19}^{19} + T_{1,7}s_{19}^{19}
= 3 \times 1 + 3 \times 4 + 1 \times 4 = 19.
\]

At this point, all of the shipping schedules needed for computing the entire production schedule for \( A_1 \) have been assembled.

8. Repeat the above steps for the next production period; in our case, repeat for production period 17. Note that the same elements of \([T]\) and \([\sigma]\) are used, and that the references to the shipping schedules are made one production period further along than was the case in steps 6 and 7.

9. Continue the iteration of steps 1 through 8 until the production schedule for \( A_1 \) has been computed to the period \( m \) representing the planning horizon. This completes the computation for article \( A_1 \). Write this schedule out.

10. Bring in the second row of \([T]\), the second row of \([\sigma]\), etc., repeating the procedures described above, starting in step 1, and developing the production schedule for article \( A_2 \). Repeat all this until the production schedules for all \( A_i \) have been completed.

**Computation of the Machine Loading Schedules, \( h_{n}^m \)**

The computation suggested by (16) is the same as a straightforward matrix multiplication, considering the production schedules \( x_{r}^m \) as a matrix of dimensions \( i, m \). This suggests storing \([r]\) by rows and \( x_{r}^m \) by columns. For several reasons, however, these machine loading schedules will be computed in a slightly different manner. The first of these reasons is that the production schedules are developed in rows (see above), are printed out in this form, and are extremely lengthy. The sorting job to be done in changing these schedules to column form is an enormous task in many practical examples. Thus, we will suggest here a scheme which makes use of each element of \( x_{r}^m \) as it appears in row sequence. The second reason for adopting the computing scheme to be suggested here is that the contents of \([r]\) are available in column form in the lists used by production people. Thus, if we can use \([r]\) in columns, rather than sort the elements into rows, we will save some effort. The third reason we consider this scheme is that the dimensions of the \( h_{n}^m \) “matrix” are reasonably small, as will be seen in a later section of this paper. These dimensions are such that in many problems the entire matrix can be stored in an internal memory of 1,000 to 6,000 words. The method is as follows.

1. Assign space in the internal memory for a matrix of elements with \( n \) (max) rows and \( m \) (max) columns, all elements containing the initial value zero.
2. Bring in the first column of \([r]\).
3. Bring in the first schedule, or “row,” of \( x_{r}^m \).
4. Multiply the first element of the first row of \( x_{r}^m \), i.e., \( x_{1}^1 \), by the elements of the first column of \([r]\), i.e., \( r_{n,1} \), adding each product into the \( h_{n}^m \) matrix at the appropriate address of the first column, \( n,1 \). This is demonstrated in the example of Fig. 12, where \( x_{1}^1 \) is seen to appear only in the elements of the first column of \( h_{n}^m \), multiplied successively by the elements of the first column of \([r]\).
5. Repeat step 4, multiplying the second element of the first row of \( x_{r}^m \), i.e., \( x_{2}^1 \), by the same first column of \([r]\), i.e., \( r_{n,1} \), adding each product into the \( h_{n}^m \) matrix at the appropriate address in the second column. Continue this procedure until all the elements of the first row of \( x_{r}^m \) have been considered.
6. Bring in the next column of \([r]\) and the next row of \( x_{r}^m \), and repeat the above procedure.
7. Repeat the above iteration until all columns of \([r]\) and all rows of \( x_{r}^m \) have been exhausted. Read out the schedules \( h_{n}^m \) as rows of the matrix which have been developed in the internal storage.

For the computation of labor requirements, a matrix of the same type as \([r]\) is provided, where the elements state the total number of man-hours of type \( n \) required to construct one article \( A_i \). The \( n \) dimension of such a labor requirements matrix is the same order of magnitude as the \( n \) dimension of the machine-hour requirements matrix. The computation would be carried out in the same manner as described here for machine loading.

**Verification Procedures**

Several procedures for verifying accuracy will be included in the present exercise. The checks included here will serve to indicate the type of check which can be performed, and their inclusion will make the time and space estimates more practical.

**Input Checks**

To maintain a check on the introduction of changes in any of the schedules and matrices, we will include checks...
to monitor the accuracy of the steps performed in the input translations. The purpose of these checks will be to detect errors in the initial keyboard entry, translation of part numbers, and actual modification of the data. These checks require that certain total quantities be generated by the personnel initiating the changes, so that the machine procedures will have something to check against. Briefly, the quantities required and procedures to be followed are as follows:

**Changes in \([N]\).**—Parts lists changes are to be accompanied by a new total number of parts represented by the assembly parts list. Note that the total number of parts in the assembly parts list for an assembly \(A_p\) is represented by the sum of the elements of the column for \(A_p\) in \([N]\). The sums of all columns will be carried with the stored \([N]\), forming an additional row. When modifications of \([N]\) are made, each column will be checked to insure that the sum of the new set of elements agrees with the new total which was supplied along with the changes.

**Changes in \([\sigma]\).**—When the setback structure is changed by the production department, the new set of setbacks for a given \(A_s\) is to be accompanied by a simple sum of the quantities stating the setbacks. When the changes have been recorded in the stored form of \([\sigma]\), the sum of the columns will be compared with these totals, in the same manner as \([N]\) is checked.

**Changes in \(s^*\).**—For changes in the shipping schedules, it will be assumed that checks on the inputs will be performed as part of other (probably daily) procedures, and hence this group of input checks will not be considered here. (These include checks on the changes caused by new orders, order changes, and shipments of completed assemblies.) It will be assumed that the total \(A_s\) on each order and the cumulative shipping schedules are stored and are available for subsequent checks.

**Changes in \([r]\).**—Here total machine hours (of all types) for each \(A_s\) should be supplied. Each column of \([r]\) will be compared with the new totals supplied with the changes.

**Checks on Accuracy of Computation**

The check which will be used repeatedly makes use of a check column or check row, or both, in the matrices and schedules employed here. Appendix I describes the principle employed.

**Check on Computation \([T]\).**—Here we will augment \([N]\) with an additional row. The contents of this row are simply the sum of the number of parts going directly into each \(A_p\). (Note that these are the same quantities used to verify the accuracy of changes introduced into \([N]\).) This will generate an augmented total requirements matrix \([T^*]\). The elements of the last row of \([T^*]\) must be the sum of the elements of the column above each. This check will be made when \([T^*]\) is complete. A run which reads all of the rows of \([T^*]\) will be made, adding each element of a given row into an address determined by the column \(k\). By the time the last (check) row of \([T^*]\) is reached, the sum of the elements of each column of \([T]\) will be complete in the storage, and these sums can be compared with the elements of the last row of \([T^*]\).

**Check on Computation of \(x^*\).**—The principle of Appendix I can be applied here in a slightly modified form. The checking technique must be changed a little to take into account the fact that each series of products, developing a schedule for one \(A_s\), does not make use of the entire contents of the schedules \(s^*\); a section of each schedule is used, starting from the \(m\) representing the setback for \(A_s\) going into that \(A_s\), and ending at a period equal to the last production period computed plus the setback. Cumulative schedules \(S^*_m\) will be stored. As the \(x^*_m\) schedule is computed for one \(A_s\), the difference between the cumulative quantities at the beginning and end of the section of the shipping schedule will be taken. This will be done for all the \(A_s\) involved, producing in effect an extra last “column” of \(s^*\). This “check column,” multiplied by the rows of \([T]\), will produce a check column in the resulting \(x^*_m\). The elements in this latter column must be the sum of the elements of each schedule \(x^*_m\) for one \(A_s\).

**Check on Computation of \(k^*_m\).**—Here a more complete check will be carried out. The matrix \([r]\) will be augmented with an extra row containing the sum of the machine-hours of all types \(n\) required to make each \(A_s\). Note that this is a quantity which can be supplied by the people developing the production techniques, and this is the quantity used in checking the accuracy of changes put into \([r]\). The “matrix” \([x]\) has an augmenting “column” derived when the \(x^*_m\) schedules are derived. The resulting augmented \(k^*_m\) will contain an extra row and column which will contain the sums of the appropriate rows and columns of \(k^*_m\). With horizontal and vertical checks, an error in one of the elements should be accompanied by check failures in one of the horizontal and one of the vertical checks. These two failures give the co-ordinates of the incorrect element, making it possible to program for a recomputation of that element alone.

**Periodic Data-Processing Procedure**

With the background of the preceding sections, we can now outline the procedure to be performed each time the production schedule, machine load requirements, and labor requirements are to be developed.

1. Introduce the changes in \([N]\). Working from the assembly parts lists and check sums, make a run through the rows of \([N]\) replacing elements with new quantities, introducing new assemblies and deleting others. This reflects the changes which have taken place in the company products since the last computation. Perform a second run through \([N]\), checking the control sums.

2. Compute the total requirements factor matrix \([T]\), as outlined above in the section on computation. This produces a complete matrix stored in columns, and a reduced version containing only the columns for shippa-
3. Sort the rows of \([T]\) computed in step 2 into ascending sequence on \(i\), so that they will be available in the proper sequence for performing the production requirements computation.

4. Introduce the changes in \([\sigma]\). As products are added or dropped, as engineering or production technique changes are made, and as experience is gained, the values of the setbacks will change. The changes are introduced and verified in the same manner as the changes in \([N]\). Since the changes in \([\sigma]\) may well be developed by different personnel than those developing the changes in \([N]\) (and hence, \([T]\)), it is entirely possible that inconsistent changes will be introduced into \([N]\) and \([\sigma]\). For example, the affects of a new assembly may pass through the paper mill leading to the changes in \([N]\) more rapidly than changes go through the mill leading to changes in \([\sigma]\). Thus, when the computation of production schedules is carried out, certain elements of \([T]\) may find no corresponding elements in \([\sigma]\). If it is not convenient to check these two sets of changes outside of the data processor, it is strongly recommended that a special consistency check run be made in the data-processing procedures. This run might compare the two sets of inputs, or, better, compare \([T]\) and \([\sigma]\) to make sure that each has nonzero elements in the same locations.

5. Consolidate shipping schedules. The organization of the actual schedules may be somewhat different from that required for \(s_{nm}\). For the purposes of this computation, we want all orders for article \(A_k\) to be consolidated into one schedule. The actual schedules may be broken into several schedules for each \(A_k\), the separate schedules being for different customers, or different destinations, or both. For this exercise, assume that such is the case, and that it is necessary to perform a consolidation of several schedules to produce the desired \(s_{nm}\).

6. Compute the production requirements schedules, \(x_{nm}\), as outlined above in the section on computation.

7. Print out the production requirements schedules. The schedules developed in the computation of step 6 are in a suitable sequence for print-out. Some editing will be necessary to arrange a suitable format.

8. Introduce the changes in \([r]\). The modification and checking procedure for making changes in the machine hour requirements matrix is similar to that described above for \([N]\) and \([\sigma]\). The program will be somewhat different here, because \([r]\) is stored in columns, while the \([N]\) and \([\sigma]\) are stored in rows.

9. Compute the machine loading schedules, \(h_{nm}\), using the general program stated above under computation.

10. Print out the machine loading schedules. The schedules developed in step 9 are in suitable sequence for print-out. Some editing will be necessary to arrange a suitable format.

11. Introduce the changes in the matrix of type \([r]\) representing labor man-hour requirements, as in step 8.

12. Compute the labor loading schedules, as in step 9.

13. Print out the labor loading schedules, as in step 10.

Magnitude of the Problem

The size and content of the various matrices and schedules involved in this exercise may be measured by the quantities defined in Table I. The range of values given in this table serve to illustrate the size of problems encountered in practice and which appear to be of sufficient magnitude to call for the services of high-speed data-processing equipment. A general picture of the range of dimensions of the several matrices and schedules is given in Table II.

Table III gives the parameters for three production control examples which have been encountered by the authors.

Consider the storage space required for storing the matrix \([N]\).

Case 1

If all elements of \([N]\) are stored, if actual part numbers are used (often requiring the storage of characters other than digits), if each row is stored by listing \(i\) once at the beginning of each row, identifying the \(j\) by the position of each element in the row, then the number of digits and characters stored is

\[(Q_1^pD_N \text{ (digits)} + Q_1C_p \text{ (characters)}).\]

Digits and characters are separated here so that they may be translated into actual storage space in different types of processors. One processor may be constructed so that it can handle alphabetic characters and other symbols internally, with all digits and characters occupying the same amount of storage space; another processor will require the use of two digit spaces for storing one alphabetic character. The storage, in digits and characters, required for \([N]\) in the three examples is shown in the first row of Table IV. Since this represents the order of 20,000 feet, five million feet, and fourteen million feet of magnetic tape, respectively (see below for the assumptions made concerning magnetic tape storage density), it appears to be advisable to consider means for reducing the storage required.

Case 2

A considerable saving in storage space can be effected if we take advantage of the fact that most of the elements of \([N]\) are zero. Consider now the storage space required if we store only nonzero elements. We will store \(i\) at the beginning of a row, and store \(j\) with each element. The storage required is

\[Q_1q_1(D_N \text{ digits} + C_p \text{ characters}) + Q_1C_p \text{ characters.}\]

The resulting storage for our examples is given in the second row of Table IV.
TABLE I  
SUMMARY OF QUANTITIES REQUIRED TO MEASURE THE SIZE OF THE COMPUTATION TASK

<table>
<thead>
<tr>
<th>Quadrant</th>
<th>Quantity</th>
<th>Range of values</th>
</tr>
</thead>
<tbody>
<tr>
<td>A4</td>
<td>number of different A4</td>
<td>$1,000 \leq Q_4 \leq 100,000$</td>
</tr>
<tr>
<td>S</td>
<td>number of machine-hours required to make one A4</td>
<td>$5 \leq Q_4 &lt; 1,000$</td>
</tr>
<tr>
<td>R</td>
<td>number of different types of machine-hours (r)</td>
<td>$10 \leq Q_4 &lt; 1,000$</td>
</tr>
<tr>
<td>T</td>
<td>average number of different A4 required to make one A4</td>
<td>$5 \leq Q_4 \leq 100$</td>
</tr>
<tr>
<td>S</td>
<td>average number of different A4 appearing in assemblies of all types and at all levels of the Gozinto graph</td>
<td>$10 \leq Q_4 \leq 10,000$</td>
</tr>
<tr>
<td>R</td>
<td>average number of different types of machine-hours required to make an assembly A4</td>
<td>$5 \leq Q_4 \leq 100$</td>
</tr>
<tr>
<td>T</td>
<td>the number of levels of the Gozinto graph</td>
<td>$2 \leq Q_4 \leq 100$</td>
</tr>
<tr>
<td>N</td>
<td>number of levels of a A4 required to make one A4</td>
<td>$0 \leq N_{c,p} &lt; 1,000$</td>
</tr>
<tr>
<td>D</td>
<td>number of digits required to store one element of [N]</td>
<td>$2 \leq D_N \leq 3$</td>
</tr>
<tr>
<td>T</td>
<td>total number of A4 required to make one shippable article A4</td>
<td>$0 \leq T_{c,b} &lt; 10,000$</td>
</tr>
<tr>
<td>D</td>
<td>number of digits required to store one element of [T]</td>
<td>$2 \leq D_T \leq 4$</td>
</tr>
<tr>
<td>S</td>
<td>number of A4 to be shipped in production period m</td>
<td>$0 \leq S_p &lt; 100,000$</td>
</tr>
<tr>
<td>D</td>
<td>number of digits required to store the designation of a period.</td>
<td>$3 \leq D_n \leq 4$</td>
</tr>
<tr>
<td>S</td>
<td>cumulative number of A4 to be shipped by production period m</td>
<td>$0 \leq S_{a,n} &lt; 1,000,000$</td>
</tr>
</tbody>
</table>

TABLE II  
SUMMARY OF QUANTITIES REQUIRED TO MEASURE THE SIZE OF THE COMPUTATION TASK

<table>
<thead>
<tr>
<th>Quadrant</th>
<th>Quantity</th>
<th>Range of values</th>
</tr>
</thead>
<tbody>
<tr>
<td>A4</td>
<td>the total setback of an A4 going into A4</td>
<td>$0 \leq s_{a,n} &lt; 10,000$</td>
</tr>
<tr>
<td>S</td>
<td>the number of A4 to be started in production period m.</td>
<td>$0 \leq r_{n,m} &lt; 100,000$</td>
</tr>
<tr>
<td>D</td>
<td>the total number of machine-hours of type n required to make one A4.</td>
<td>$0 \leq h_{n,m} &lt; 10,000$</td>
</tr>
<tr>
<td>C</td>
<td>number of characters required to store a standard part number.</td>
<td>$5 \leq C_p \leq 24$</td>
</tr>
<tr>
<td>D</td>
<td>number of digits required to store a standard part number used within the data processor.</td>
<td>$3 \leq D_p \leq 5$</td>
</tr>
<tr>
<td>D</td>
<td>number of digits required to store a check sum for one column or row of matrix ().</td>
<td>$4 \leq D_{1,1} \leq 7$</td>
</tr>
<tr>
<td>C</td>
<td>number of characters required to store a standard machine-type designation n.</td>
<td>$5 \leq C_n \leq 10$</td>
</tr>
<tr>
<td>D</td>
<td>number of digits required to store a translated machine-type designation n used within the data processor.</td>
<td>$2 \leq D_n \leq 3$</td>
</tr>
</tbody>
</table>

While these requirements are more reasonable than in Case 1, it still appears to be highly desirable to reduce the storage requirements further, if possible. The example considered here, [N], is but one of the several matrices and schedules involved in this problem. Any small per cent reduction in storage requirements will buy a very real saving in storage medium cost.

Case 3
Consider now the use of two sets of part numbers. A set of part numbers will be established for internal use throughout the computation. A translating table must be carried, giving the part numbers used by people and the corresponding part numbers used internally. Now, [N] requires

$$M_n = Q_4 q_4 (D_N + D_p) + Q_4 D_p \text{ digits.} \quad (18)$$

The translating table occupies the following storage:

$$M_{T_{1,1}} = Q_4 (C_p \text{ characters} + D_p \text{ digits).} \quad (19)$$
TABLE III
EXAMPLES OF PRODUCTION CONTROL PARAMETERS

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Q_1)</td>
<td>3,000</td>
<td>50,000</td>
<td>80,000</td>
</tr>
<tr>
<td>(Q_2)</td>
<td>100</td>
<td>250</td>
<td>200</td>
</tr>
<tr>
<td>(Q_3)</td>
<td>18 months</td>
<td>12 months</td>
<td>24 months</td>
</tr>
<tr>
<td>(Q_4)</td>
<td>50</td>
<td>100</td>
<td>250</td>
</tr>
<tr>
<td>(Q_5)</td>
<td>10</td>
<td>15</td>
<td>20</td>
</tr>
</tbody>
</table>

The figures in row 3 of Table IV show that this maneuver reduces the storage space for \([N]\) to about half that required in Case 2. Row 4 shows that the translating table for part numbers occupies from 10 per cent to 20 per cent of the space required for \([N]\). It is to be noted that the storage space required for the translating table is not to be charged solely to the reduction in storage space for \([N]\). This single table effects reductions in storage space for all the matrices and schedules having one or more dimensions represented by part numbers.

The disadvantages of using an internal part number in addition to the external numbers are (1) storage space is required for a translating table, (2) all inputs and outputs connecting with people must be translated, and (3) the translation requires time and introduces the possibility of errors in translation. The errors in input translation will be detected by the input checking procedures described above. Similar checks on outputs might be instituted, checking control totals after the translation to external part numbers. The advantages of using short internal part numbers include (1) considerable net saving in storage space, (2) great savings in computation time, due to the fact that most of the processing can be carried on with the internal part numbers, using much shorter lengths of storage medium, and (3) easier comparison of part numbers during the body of the processing, especially if the machine is a fixed word length processor and the part numbers are greater than one word in length.

The storage space required to carry the checking column for \([N]\) is:

\[
M_{Nech} = Q_1(D_{Nech} + D_p).
\]  

Assigning 5 digits to \(D_{Nech}\) for example A, and 6 digits for examples B and C, we obtain the digit storage requirements for checking, given on the fifth row of Table IV. The space required varies from 7 per cent to 14 per cent of the storage space required for \([N]\).

TABLE IV
STORAGE REQUIRED FOR VARIOUS ORGANIZATIONS OF \([N]\)

<table>
<thead>
<tr>
<th></th>
<th>Example A</th>
<th>Example B</th>
<th>Example C</th>
</tr>
</thead>
<tbody>
<tr>
<td>(D_p)</td>
<td>.027X10^6</td>
<td>.07X10^6</td>
<td>.14X10^6</td>
</tr>
<tr>
<td>(C_p)</td>
<td>30</td>
<td>1,100</td>
<td>1,500</td>
</tr>
</tbody>
</table>

\(D=\)Digits, \(C=\)Characters, \(FT=\)Feet of magnetic tape.

The magnetic tape storage requirements are based on an assumed 100 characters per inch, with 75 per cent of the tape length containing information. The figures stated assume storage of data at the full density possible. In practice, considerations such as fixed word length, fixed block length, ease of programming, etc., will reduce the storage efficiency and increase the storage tape length required.

The major storage areas required for this problem are summarized in Table V. The figures given for matrices and schedules include check rows and columns. Here again, actual storage requirements will be considerably larger than the quantities quoted, for reasons such as those given at the end of the previous paragraph.

**PROCESSOR CHARACTERISTICS**

The above sections have defined the type of computation required and the volume of data involved in typical examples. This information makes it possible to state certain data-processor characteristics which are neces-
sary or highly desirable in processors applied to the work described here. A general method of performing the data-processing task has been suggested. This method has been postulated keeping in mind the general characteristics of data-processing equipment now available or soon to be available. The method described does not, however, confine itself to the abilities of existing equipment; indeed, the method described is probably not wholly feasible on existing equipment, whose limitations will usually dictate departures from the direct method described here. Such departures will nearly always increase the time required to perform the computation. As a result of these considerations, a few desirable characteristics suggest themselves. These are given below. The fact that some of the desirable characteristics stated below are not available in existing equipment does not preclude the performance of this type of problem on available processors. The matters suggested below are given to point out characteristics which will sharply increase the speed with which problems of this type and size can be performed on automatic data-processing equipment.

### TABLE V

**Storage Required for the Major Parts of the Problem**

<table>
<thead>
<tr>
<th></th>
<th>Millions of characters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Example A</td>
</tr>
<tr>
<td>[N]</td>
<td>.22</td>
</tr>
<tr>
<td>Part number translation</td>
<td>.063</td>
</tr>
<tr>
<td>[T]</td>
<td>.15</td>
</tr>
<tr>
<td>List of shippable articles $A_0$</td>
<td>.0013</td>
</tr>
<tr>
<td>$s_k^m$</td>
<td>.043</td>
</tr>
<tr>
<td>$x_k^m$</td>
<td>.15</td>
</tr>
<tr>
<td>$r_m$</td>
<td>.56</td>
</tr>
<tr>
<td>Machine type translation</td>
<td>.087</td>
</tr>
<tr>
<td>$h_k^m$</td>
<td>.004</td>
</tr>
<tr>
<td></td>
<td>.0043</td>
</tr>
<tr>
<td>Total</td>
<td>1.26</td>
</tr>
<tr>
<td></td>
<td>1,400 FT</td>
</tr>
</tbody>
</table>

**Type of Characters**

The processor should be capable of processing alphanumeric characters, plus several other symbols encountered in part numbers (including dash, slash, and blank). This capability is necessary if the part "numbers" used internally are the same as those used by people. If the part numbers are translated into shorter numerical part numbers, as is strongly suggested above, it is still highly desirable to be able to handle nondigital characters within the machine, so that the processor can bear the burden of the attendant translating tasks.

**Word Length**

A variable word length machine can be used to some advantage here, providing a small advantage in storage efficiency compared with fixed word length storage, and providing a considerable reduction in programming maneuvers required to extract and shift parts of words.

If a fixed word length machine is to be used, we can get some idea of the desired word length by considering the storage organization suggested in the section on magnitude of the problem. In general, the data in the various matrices and schedules is stored with each element accompanied by one co-ordinate. On this basis, for nearly all of the storage required, the examples given here and the worst cases to be expected in practice require no more than 10 characters per word. One of the exceptions is the set of schedules $s_k^m$. Here we postulated the storage of accumulative shipping quantities along with the quantity per period and the period designation $m$. In two of the examples, this calls for 12 characters, and up to 15 characters are indicated for larger problems. Since this is but one part of the total storage problem, it should not be given great weight. For $s_k^m$, two adjacent words may be required for each element. Finally, the translating tables for part numbers, machine-type designations, and types of labor would often require very lengthy words for the numbers and designations used by people. There exist part numbers up to 25 characters in length. A widely encountered part number length is the 14-character Army-Navy number. Such lengthy numbers can be fitted into several shorter fixed-length words in the translating tables.

**Fixed or Floating Point**

For the purposes of the computation described here, a fixed point machine is satisfactory. In any given factory, the range of magnitude of the elements of each matrix or schedule is well known, much of the work involves small integers, and the range of the intermediate and final results is well behaved.

**Form and Size of Major Storage**

The figures at the bottom of Table IV give very rough measures of the total storage required in this work. For the largest example shown, storage of the order of ten to fifty million characters is required. Of the forms of storage available at present, magnetic tape appears to be the only one suitable for problems of this magnitude, for reasons of both volume of storage medium and input-output time.

A very general idea of the number of tape reels which should be electrically connected to the processor can be gained by considering one part of the process, such as the development of the $[T]$ matrix. During this computation repeated runs are made through $[N]$ and $[T]$. There are as many passes through $[N]$ as there are levels in the Gotozino graph, or $q_2$ passes. There are as many passes through $[T]$ as there are articles $A_i$, or $Q_i$ passes. By the end of the computation of $[T]$, it occupies more storage space than $[N]$ by the ratio of $q_2$ to $q_1$. In addition, the reduced form of $[T]$, containing only columns for shippable articles of $A_i$, has been developed and written out. If these repeated passes
through the tapes are interrupted by manually changing reels thousands of times, the efficiency of the operation will be seriously impaired. It is desirable that magnetic tape storage capable of handling \( [N] \) and both versions of \( [T] \) be available.

Assume a tape storage density of 100 cells per inch, with 75 per cent of the tape length actually containing blocks of information. Assume further that the digit storage requirements stated in Table IV are placed in words and blocks in such a way that an 80 per cent storage efficiency is attained. Under these conditions, the tape storage desired in these examples is: A, 100 feet; B, 20,000 feet; C, 55,000 feet. Taking into consideration the fact that separate sets of data should be on separate reels (i.e., \( [N] \) and the two versions of \( [T] \) should be separated), the number of 2,500-foot reels indicated is: A, 3; B, 10; C, 23.

The order of 11 to 20 hoppers of some kind (e.g., tape reels, tape loops) is desirable for ease and efficiency of sorting.

**Internal Storage**

A very rough programming of this work and time estimates based on the present state of the art indicate that an “immediate access” storage of the order of 6,000 ten-digit words is the minimum desirable for problems of the magnitude of examples B and C. This should be backed up by a cyclic memory of the order of 20,000 words.

**Instructions**

The extensive matrix multiplication type of computation encountered in this problem argues for a multiply instruction which accumulates the sum of a series of products.

The performance of this problem includes several table look-up operations. An example of this takes place during the computation of \( [T] \), where complete rows of \( T_{p,j} \) are developed during the computation, but only the elements falling in columns representing shippable articles are to be written out in row form. Thus, the part number \( j \) for each element \( T_{p,j} \) will be compared with a list of part numbers of shippable articles. If a storage with a 5-millisecond average access time is used to store the table (e.g., a 6,000 rpm magnetic drum with one head per band) the table look-up time in the three examples would be: A, 25 minutes; B, 17 hours; C, 25 hours. This is one of the lesser table look-up tasks in this problem, and even the times quoted here would jump greatly if the number of shippable articles were increased by including large numbers of shippable spares. It is highly desirable to cut this time by two orders of magnitude, from the 5-millisecond average access to 50 microseconds. This suggests that storage space for such tables be of the “immediate access” type if possible. If the table is stored in such a memory or in a cyclic memory, a “table look-up” instruction or its equivalent is strongly indicated here. In cyclic memories, this instruction should be mechanized to compare the compare with each word as it passes the reading station, starting at a specified address.

If a fixed word length machine is used, and matrix and schedule information is stored as specified here, a good part of the internal programming will be devoted to extracting a part of a word for comparison with another word or part of another word. This sort of thing will be performed continuously in determining the co-ordinate of each element as matrix multiplications are being performed, as certain columns of \( [N] \) are sought in the computation of \( [T] \), etc. This suggests the need for a comparison instruction which includes a means for stating which section of a word is to be compared. Similar arguments apply to other arithmetic and logical instructions performed in this work. Considerable time will be saved and programming details will be appreciably reduced if addresses in all instructions include a means for stating the beginning and end of any section of a word desired.

**Output Printing**

The schedules printed out are the production requirements schedules, \( x_{n}^{*} \), the machine loading schedules, \( h_{n}^{*} \), and the labor loading schedules, similar to \( h_{n}^{*} \). Assuming a 120-column form with the identifying part number, machine-type number, or labor type number printed on the left, followed by the quantities in the schedules spread across the page with 3 columns separation between quantities, and using as many lines as necessary to state a complete schedule for each part number, etc., the following modest printing loads are estimated for the examples: A, 6,000 lines per week; B, 50,000 lines per month; C, 160,000 lines per month.

**Appendix I**

**Verification of the Accuracy of Matrix Multiplication**

Consider:

\[
[A] \cdot [B] = [C]
\]

Augment \( [A] \) with an extra row along the bottom, composed of the sum of the elements of each column. These are \( \sum a_{i,j} \). The dimensions of the resulting \( [A^{*}] \cdot [B] = [C^{*}] \), are

\[
(i + 1, j)(j, k) = (i + 1, k).
\]

So \( [C^{*}] \) has an extra row along the bottom.

\[
c_{i,k} = \sum_{j} a_{i,j} b_{j,k}
\]

\[
c_{i,k}^{*} = \sum_{j} a_{i,j}^{*} b_{j,k}.
\]

The last row of \( [C^{*}] \), \( i = m + 1 \), is

\[
c_{m+1,k}^{*} = \sum_{j} a_{m+1,j}^{*} b_{j,k} = \sum_{j} \left( \sum_{i} (a_{i,j}) b_{j,k} \right) = \sum_{j} \sum_{i} a_{i,j} b_{j,k}
\]
Thus, the elements of the last row of $[C^*]$ are the sums of the elements of the columns above the elements of $c^*_{m+1,k}$.

Example 1

Using a check row in $[A^*]$, producing a check row in $[C^*]$: 

\[
\begin{align*}
0 & 1 & 0 & 3 & 1 & 0 & 0 & 1 & 4 & 5 & 6 & 3 \\
0 & 0 & 2 & 1 & 1 & 2 & 3 & 0 & 1 & 1 & 3 & 1 \\
3 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 4 & 1 & 1 & 4 \\
1 & 2 & 1 & 0 & 1 & 1 & 1 & 1 & 3 & 4 & 7 & 1 \\
4 & 3 & 3 & 5 & 12 & 11 & 17 & 9
\end{align*}
\]

By augmenting $[B]$ with an extra column on the right, composed of the sums of the elements of each row of $[B]$, an extra check column of $[C^*]$ can be generated. Now, an erroneous element in $[C]$ can be located by noting the row and column in which the sums do not check in $[C^*]$.

Example 2

Using a check row in $[A^*]$ and a check column in $[B^*]$, producing a check row and column in $[C^*]$: 

\[
\begin{align*}
0 & 1 & 0 & 3 & 1 & 0 & 0 & 1 & 2 & 4 & 5 & 6 & 3 & 18 \\
0 & 0 & 2 & 1 & 1 & 2 & 3 & 0 & 6 & 1 & 1 & 3 & 1 & 6 \\
3 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 4 & 1 & 1 & 4 & 10 \\
1 & 2 & 1 & 0 & 1 & 1 & 1 & 1 & 3 & 4 & 7 & 1 & 15 \\
4 & 3 & 3 & 5 & 12 & 11 & 17 & 9 & 49
\end{align*}
\]

Application of Data Processors in Production

C. R. DeCARLO†

The past year has seen the firm establishment of the large electronic computer in industry. With the delivery and use of eighteen IBM 700 series of Electronic Data Processing Machines, as well as the delivery of machines by other suppliers, the electronic data processor has found a large measure of acceptance among American business management.

These large computers were originally conceived, developed, and installed as scientific or engineering aids. Indeed they have been of invaluable assistance in the aircraft and oil industries, in government scientific laboratories, particularly those dealing in nuclear power and weapons, as well as playing a significant part in the solution of important logistical problems.

As the data processing and computing capabilities of these machines became known to an ever-increasing number of people outside the scientific field, it was natural that attempts should be made to utilize them in solving problems in production and accounting.

However, the large electronic data processor, with its high arithmetic speed, capacious storage, and high-speed reading and writing, poses not only many opportunities, but also some problems in its application to production and accounting functions. Questions arise concerning the precise definition of logic used in a particular application. The availability and validity of data necessary to a machine operation are not always guaranteed. Often present departmental responsibilities must, or should, be consolidated in order to extract maximum efficiency from the data processor. These and many other problems require wisdom, mature judgment, and much patience on the part of the management group first approaching electronic data processing.

Parallel with the development and use of the electronic computer, and its appreciation by production managers, there has been developing a body of knowledge concerning the fundamental nature and theory of the production process.

This has been brought about largely through the efforts of mathematicians and production experts attracted to theoretical considerations; perhaps by the existence of the large-scale computer—a device which can serve as a laboratory to test their theories, as well as a tool for the ultimate implementation of their work.

Already several interesting and important applications of large electronic data processors have been made to particular production problems. One of these occurs in the aircraft industry, where a customer is using the IBM Type 701 EDPM for production scheduling and the control of shop orders.

The origin of any problem in manufacturing control is the production forecast, sales forecast, or contract commitment. These tell what must be made and when. From this will follow the computation of gross requirements, net requirements after processing with inventory files, and capacity requirements after scheduling.

In the case of an airplane manufacturer, the procurement of a contract is the starting point. A master

† International Business Machines, New York, N. Y.