Simultaneous Manufacturing in Batch Production

Ana Almeida*, Carlos Ramos*, Silvio do Carmo Silva†
email: ana@dei.isep.ipp.pt, csr@dei.isep.ipp.pt, scarmo@dpi.uminho.pt

*Instituto Superior de Engenharia do Porto, GECAD - Knowledge Engineering and Decision Support Research Group & Dept. Engenharia Informática
Rua São Tomé, 4200 Porto – Portugal, Phone: +351- 228340500
†Universidade do Minho, Dept. de Produção e Sistemas
Gualtar, Braga – Portugal, Phone: 351- 253604100

Abstract

This paper is concerned with the implementation of Simultaneous Manufacturing philosophy in batch production of job-shop like manufacturing systems. It addresses the processing of jobs, either simple, requiring the manufacture of a batch of parts, i.e. simple products, or complex, comprehending the parts fabrication and their multistage assembly into a batch of products. In this work the simultaneous manufacturing philosophy was implemented through the widespread use of batch overlapping, which proved particularly effective in reducing job throughput time, maintaining operating simplicity and requiring reduced coordination.

1 Introduction

Our approach considers a job to be a manufacturing order for a number of identical products. These may be simple, i.e. having a set of operations to be carried out in a given sequence in a batch of identical parts, called simple products. The products may also be complex products, having a set of operations carried out in a given sequence in a batch of products, each one made of several different parts. The manufacture of these complex products is typical of industries of type A and T as defined in [1]. It is mainly, but not only, for these types of industries that the mechanism presented in this paper is most appropriate. A job is specified by the quantity of products necessary, i.e. its batch size, the manufacturing operations and their precedence relationship, processing times of operations and the manufacturing processors to be used. Processing times include auxiliary time elements such as set up time and time for parts handling at a machine. The most generic job requires several operations on identical and different parts followed by assembly operations. A typical job is, for example, the manufacture of an ordered quantity of identical chairs. The manufacture of a certain amount of a specific component of a chair is not a job.

We strongly explore the possibility of different batch overlapping which is easy to implement and can be of great impact in reducing job flow times, in any manufacturing processors setting, when alternative manufacturing processors exist batch splitting may become a specially attractive strategy for shortening flow times. Actually, we focus the shortening of job throughput times in order to meet due dates. The reason is that this has a positive contribution to both profits and customer service.

The scheduling approach combining all or some of these factors for reducing job throughput time has been named as Simultaneous Manufacturing [2].

2 Simultaneous Manufacturing

The Simultaneous Manufacturing (SM) philosophy aims at the complete manufacturing of each single product of a product order in the minimum possible time. The intention is to take the minimum time to manufacture the whole job, i.e. the product order. To achieve this, each set of parts belonging to each product of the order must flow in a coordinated way through the system, i.e. in a way such that they are processed before any other parts, and arrive simultaneously to where they are needed for assembly. This assembly must be performed immediately, according to available processors. In this way, the throughput time for the complete manufactured and assembly of each product of the product order is minimum, and therefore the full job throughput time is minimum too. Additionally, the work-in-process is likely to be low. As we can envisage SM calls for intensive batch overlapping and for batch splitting under alternative processors. SM philosophy does not guarantee the most economical way of manufacturing and order. In fact, under simultaneous widespread use alternative processors by the same product order, its throughput time may be very small indeed, but set-up costs tend to be high. In our scheduling approach, mechanisms were developed for implementing SM in a user-controlled manner.
3 Batch Production

In traditional batch production, a job is considered as a set of identical parts that are always processed as a whole, i.e. the full batch must be processed in a processing stage before it can be transferred to another to carry out further processing. In cases where the batch size is large, this can become too great a penalty to the full duration of the job processing. So the performance of the manufacturing system, mainly regarding job throughput time and accomplishment of job due dates can become highly poor. This operating weakness can be highly reduced through batch overlapping and batch splitting. We explore these strategies in our approach implement the Simultaneous Manufacturing philosophy.

3.1 Batch Splitting

Batch splitting means partition of a batch into several smaller batches to be processed independently on machines. In this way, a substantial reduction on average batch flow time may be achieved. This may be particularly so when alternative machine or processors exist for simultaneous manufacture of split batches. An apparent disadvantage of this strategy is the overall increase of batch set-up times. This may not be important, neither be pernicious to job flow times, if processors, which are set-up, are not critical, i.e. are not bottlenecks.

3.2 Batch Overlapping

Batch overlapping means transferring work from a machine, which is processing an operation of the job, to another machine, for processing the next operation, before the entire batch has been finished on the previous machine. This is very common, in practice, sometimes done randomly, with different amounts of overlapping, and other times under a well defined overlapping procedure. In this case, transfer batches are clearly defined. These are batches, normally smaller than the total job batch size, transferred between two successively required machines. When a transfer batch is equal to the total job batch size batch overlapping does not take place. In the extreme, when trying to fully implement Simultaneous Manufacturing we should seek maximum overlapping, i.e. the transfer of work between processors should be continuous, which means transfer batches of size one.

Batch overlapping does not necessarily changes the processing batch size at a processing stage. A processing batch size is the amount of units of a job processed in a machine continuously before it takes another job.

Batch overlapping requires combination with batch streaming. This means that batch processing and batch transfer should be linked in a manner that each transfer batch can be transferred to its successor operation, immediately upon completion. Therefore, several transfer batches can be concurrently processed at different job operations. This guarantees low job throughput time, contributing for improved product delivery, and is likely to reduce work-in-process inventory. However, splitting a batch into multiple transfer batches for overlapping, couples machines at different stages. This coupling calls for a more close control of production, eventually requiring an increased effort in rescheduling under disturbances. Additionally, if products arrive at machines in identical transfer batches, intermittent idling may exist at machines whenever a predecessor operation requires a longer processing time. In our research, we developed a mechanism to generate what we call Job Patterns, which avoid the existence of those inactive or idle periods of time.

4 Job Patterns

One of the main mechanisms, which allow implementing SM in our work, is what we call Job Patterns (JP) [3]. A JP is a manufacturing processor occupation scheme of a given job, based on both batch overlapping and/or batch splitting. A JP is one of several possible alternative occupation schemes, for a given job, under an empty manufacturing system, i.e. with all processors considered available, in an unspecified time horizon. A job can have several JP, which can be generated for better exploring the utilization of manufacturing processors or machines.

4.1 Operation Plans

A requirement for establishing a JSP is the job operation plan. This specifies all operations and their precedence interrelationships of each single product, simple or complex, of a job order. It may be expressed as graph as the one shown in figure 1. Two types of graphs can represent an operation plan namely an outree and intree graph [4]. In our work both of them are considered.

![Intree precedence graph representation of a Job operation plan](image)

Figure 1 Intree precedence graph representation of a Job operation plan

An important concept is the level of the operation, we consider the operations without precedents as operations of level one, and last level operations are those, which do not have succeeding operations. The other operations are called internal operations.
4.2 Notation used
Before present the computation procedure, the used notation is introduced.

$\text{prec}(\cdot)$ precedence operator

$\text{succ}(\cdot)$ succession operator

$a$ and $b$ represent two different subsequent /succeeding operations of the same job $k$ to be processed at different manufacturing processors, where $b = \text{succ}(a)$

$a \succ b$ represents a precedence relationship, meaning $a$ directly precedes $b$

$r_k$ job release date

$t_{p_a}$ operation $a$ processing time for one product unit

$t_{p_b}$ operation $b$ processing time for one product unit

$t_{l_{p(k)}}$ job $k$ processing batch

$t_{l(a \succ b)}$ transfer batch between $a$ and $b$

$\Delta t_{s(a \succ b)}$ displacement or overlapping time between the starting time instants of two operations $a$ and $b$

$\Delta t_{f(a \succ b)}$ displacement or overlapping time between the finishing time instants of two operations $a$ and $b$

$t_{\text{start}(a)}$ instant of processing start for operation $a$

$t_{\text{fin}(a)}$ instant of processing end for operation $a$

$t_{\text{start}(b)}$ instant of processing start for operation $b$

$t_{\text{fin}(b)}$ instant of processing end for operation $b$

4.3 Computation of JP

The computation procedure determines the starting and finishing time instants for all the operations of a job on each processor or machine depending on transfer batch size, on the job operation plan adopted and on machines to be used. This establishes a JP.

To begin we need to distinguish between critical paths and non-critical paths in the graph of the operation plan. A critical path has the longest duration of operations from the start node of the graph to the last. The computation procedure is firstly applied to critical paths and then to other paths.

Overall computing procedure
Compute the duration of each graph path and store the sequence of operations for each one –

Execute Step 1

For the critical paths
From the operation of level one until the last level operation
If $\text{prec}(op_i) = \emptyset$ and $op_i$ is the level one operation
Then compute $t_{\text{start}(op_i)}$ and $t_{\text{fin}(op_i)}$ –

Execute Step 2

store the obtained values at the list of processed operations
Else make $a = op_i$ and $b = \text{succ}(op_i)$
compute the displacement $\Delta t(a \succ b)$ –

Execute Step 3
compute $t_{\text{start}(b)}$ and $t_{\text{fin}(b)}$ –
store the obtained values at the list of processed operations

For the non-critical paths
From the last level operation until the operation of level one
If $op_i \notin$ to the highest duration graph path make $a = op_i$ and $b = \text{succ}(op_i)$
compute $t_{\text{start}(a)}$ –

Execute Step 5
compute the displacement $\Delta t(a \succ b)$ –

Execute Step 6
store the obtained values at the list of processed operations

Computing Procedure Steps

Step 1 – Computation of the processing time for all the graph paths. Ordering and storing of operations for each graph path

For each graph path, sum up the processing times of all operations:

$$tp_{C_l} = \sum_{op_i \in C_l} t_{p_{op_i}}, \quad (1)$$

where: $C_l$ with $l=1, 2, ..., c$ represents each graph path

$\text{prec}(op_i) = \emptyset \land \text{succ}(op_i) = \emptyset$

$t_{p_{op_i}}$ it is the processing time of operation $op_i$

$t_{p_{C_l}}$ it is the jobs total processing time for the graph path under consideration

The following set is obtained: $T_{p_{C_l}} = \{t_{C_l}\}$

where: $C_l$ with $l=1, 2, ..., c$ represents each graph path

Step 2 – Computation of the starting and finishing instants for the first operation in critical paths

$t_{\text{start}(a)} = r_k$ ;

where: $r_k$ its the job release date
allows defining the interruption, i.e. intermittent idling. This displacement which processes operation ensures enough feeding to the machine lowest time interval between the starting of the two adjacent operations.

The batch displacement between two succeeding adjacent operations \(a\) and \(b\) of the same job, is the lowest time interval between the starting of the two operations that ensures enough feeding to the machine which processes operation \(b\) for avoiding processing interruption, i.e. intermittent idling. This displacement allows defining the transfer batch \(lt(a \gg b)\).

Considering a job batch size of \(n\) units, the size of the transfer batch depends on factors such as the minimum and maximum amount work that can be transferred between the manufacturing processors and the buffer size of the receiving manufacturing processor or machine. The transfer batch can, therefore, be unitary or have a size not exceeding the size of the job batch size. In order to simplify the computation procedure we consider that \(n/lt(a \gg b) \in \mathbb{N}\).

Two cases can occur depending on operation processing times, so the computation mechanisms are different.

Case a) If \(tp_a \leq tp_b\)

then \(tp_b * n \geq lt(a \gg b) * tp_b + (n - lt(a \gg b)) * tp_a\)

This means that the total processing time for all the units of the batch for the operation \(b\) is long enough to ensure that no interruption occurs during batch processing at operation \(b\). So the processing of operation \(b\) can start as soon as the transfer batch arrives at the respective manufacturing processor, after the first transfer batch has been processed in operation \(a\).

Figure 2 illustrates the relationships between important variables in such a way that one can easily understand how the displacement for this case, is obtained, namely through the following expression:

\[
\Delta ts(a \gg b) = lt(a \gg b) * tp_a
\]  

\[t_{fin}(a) = t_{start}(a) + n * tp_a \quad (3)\]

Case b) If \(tp_a > tp_b\)

then \(tp_b * (n - lt(a \gg b)) < (n - lt(a \gg b)) * tp_a\)

so:

\[tp_b * n < lt(a \gg b) * tp_b + (n - lt(a \gg b)) * tp_a\]

This means that the total processing time for all the units of the batch for the operation \(b\) is not long enough to avoid interruption during batch processing. So if the processing of operation \(b\) starts as soon as the transfer batch arrives at the respective manufacturing processor, after operation \(a\) has been processed, then it would occur a batch processing interruption due to starving, i.e. intermittent idle time would take place. Therefore, the displacement must be long enough to ensure processing of operation \(b\) without interruption. Figure 3, illustrates how such displacement is obtained in this case.

\[t_{fin}(a) = t_{start}(a) + \Delta ts(a \gg b) + n * tp_a \quad (4)\]

**Step 3 – Computation of the displacement or overlapping time between two succeeding adjacent operations \(a\) and \(b\) \((\Delta ts(a \gg b))\) from critical paths**

Figure 2 Displacement representation for Step 3 when \(tp_a \leq tp_b\)

**Step 4 – Computation of the starting and finishing instants for the other operations of critical paths**

\[t_{start}(b) = t_{start}(b) + \Delta ts(a \gg b) \quad (6)\]

where \(a\) is the operation with the greater starting instant \((t_{start})\), in the universe of all the preceding operations of \(b\).

**Step 5 – Computation of the starting and finishing instants for the all the operations of the other paths**

\[t_{fin}(b) = t_{start}(b) + n * tp_b \quad (7)\]
Case a) If \( tp_a \leq tp_b \)

\[
\begin{align*}
    t_{\text{start}(a)} &= t_{\text{start}(b)} - \Delta \text{ts}(a \rightarrow b) \times tp_a \\
    t_{\text{fin}(a)} &= t_{\text{start}(a)} + n \times tp_a
\end{align*}
\]
(8)

Case b) If \( tp_a > tp_b \)

\[
\begin{align*}
    t_{\text{fin}(a)} &= t_{\text{fin}(b)} - \Delta \text{ts}(a \rightarrow b) \times tp_b \\
    t_{\text{start}(a)} &= t_{\text{fin}(a)} - n \times tp_a
\end{align*}
\]
(9)

When operation \( a \) also belong to the graph path with the highest duration its starting and finishing time instants are already know, so in this case it is not necessary to compute those values again.

**Step 6** – Computation of the displacement \( \Delta \text{ts}(a \rightarrow b) \) between the starting time instants of two succeeding adjacent operations \( a \) and \( b \), belonging to a non-critical path.

Case a) If \( tp_a \leq tp_b \)

\[
\Delta \text{ts}(a \rightarrow b) = t_{\text{start}(a)} - t_{\text{start}(b)}
\]
(12)

Case b) If \( tp_a > tp_b \)

\[
\Delta \text{ts}(a \rightarrow b) = t_{\text{start}(b)} - t_{\text{start}(a)}
\]
(13)

We adopt the following representation for a \( JP \):

\[
JP_k = \{mp_1, (op_1, t_{\text{start}}, t_{\text{fin}}), \ldots, (op_n, t_{\text{start}}, t_{\text{fin}}), \ldots, (mp_n, (op_1, t_{\text{start}}, t_{\text{fin}}), \ldots, (op_n, t_{\text{start}}, t_{\text{fin}}))\}
\]

5 **Illustrative Example**

Supposing that we intend to manufacture a batch with 8 product units within a deadline, \( td \), of 70 \( tu \), being the release date, \( rk \), the instant 0, we have: \( n = 8, \Delta d = 70 \) and \( rk = 0 \). The precedence graph is represented in figure 1. In table 1 are represented the processing times, the respective manufacturing processor for each operation, the preceding and succeeding operations as well as the occupation times for each manufacturing processor.

<table>
<thead>
<tr>
<th>( op_k )</th>
<th>( mp )</th>
<th>( tp )</th>
<th>( \text{Prec}(op_k) )</th>
<th>( \text{Succe}(op_k) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( op_1 )</td>
<td>1</td>
<td>8</td>
<td>-</td>
<td>( op_2 )</td>
</tr>
<tr>
<td>( op_2 )</td>
<td>2</td>
<td>3</td>
<td>24</td>
<td>( op_1 )</td>
</tr>
<tr>
<td>( op_3 )</td>
<td>3</td>
<td>1</td>
<td>8</td>
<td>-</td>
</tr>
<tr>
<td>( op_4 )</td>
<td>4</td>
<td>2</td>
<td>16</td>
<td>( op_2 ), ( op_3 )</td>
</tr>
</tbody>
</table>

Considering the data of the example and a unitary transfer batch, the result obtained by the job scheduling pattern generator mechanism, is illustrated in figure 3 where the batch displacement between \( op_1 \) and \( op_2 \), and between \( op_2 \) and \( op_4 \) is shown.

![Figure 3](image-url)

**Figure 3** Job Scheduling Pattern with batch displacement representation

\[
JP_k = \{(mp_1, (op_1, 0, 8)), (mp_2, (op_2, 1, 25)), (mp_3, (op_3, 10, 18)), (mp_4, (op_4, 11, 27))\}
\]

We point out that operation \( op_3 \) could start anytime between instants 0 and 10. By starting at instant 10 a late start JSP is used. If had started at instant 0 we were adopting the earliest start JSP.

If, instead of a unitary transfer batch, we consider, in the other extreme, a transfer batch with the same size of the job batch the resulting job scheduling pattern is different and shown figure 4.

![Figure 4](image-url)

**Figure 4** Job Scheduling Pattern with transfer batch equal to processing batch

\[
JP_k = \{(mp_1, (op_1, 0, 8)), (mp_2, (op_2, 8, 25)), (mp_3, (op_3, 24, 32)), (mp_4, (op_4, 32, 48))\}
\]

It can be noticed that the job flow time has increased from 27 \( tu \) to 48 \( tu \). The starting and finishing instants of the occupation periods of the processors became different although the amount of time involved is the same.
6 Conclusions

Under today’s highly competitive markets it is important to provide good service to customers and, at the same time, reduce costs in manufacturing. One clear contribution on these lines is to manufacture job orders within the shortest time that is reasonably possible, keeping work in process low [5]. This has a direct effect on profitability, since fast turnover of short-term investment is achieved, and, of course, on fast deliveries, which is an important requisite for customer satisfaction.

We presented in this paper an approach oriented towards achieving very short manufacturing throughput times of customer orders. The approach uses the Job Pattern concept, described in detail, and implements a manufacturing philosophy known as Simultaneous Manufacturing. The main imbedded strategy used can be seen as a powerful aid to improve the efficiency in complex manufacturing environments, integrating both fabrication of parts and their assembly into customer ordered products.

References


