Division of Research
School of Business Administration
The University of Michigan

December 1991

PART TYPE SELECTION AND BATCH SEQUENCING IN A FLEXIBLE FLOW SYSTEM

Alessandro Agnetis
Universita degli Studi di Roma "La Sapienza"

Claudio Arbib
II Universita degli Studi di Roma "Tor Vergata"

Kathryn E. Stecke*
The University of Michigan

Working Paper # 670

*This research was begun while the third author was at the Fraunhofer Institute für Arbeitswirtschaft und Organisation, Stuttgart, Germany.

PART TYPE SELECTION AND BATCH SEQUENCING IN A FLEXIBLE FLOW SYSTEM

Alessandro Agnetis
Università degli Studi di Roma "La Sapienza"
Dipartimento di Informatica e Sistemistica
Roma, Italy

Claudio Arbib
II Università degli Studi di Roma "Tor Vergata"
Dipartimento di Ingegneria Elettronica
Roma, Italy

Kathryn E. Stecke*
The University of Michigan
Graduate School of Business Administration
Ann Arbor, Michigan

December 1991

*This research was begun while the third author was at the Fraunhofer Institute für Arbeitswirtschaft und Organisation, Stuttgart, Germany.

ABSTRACT

A new approach to part type selection and the scheduling of batches over time in a flexible flow system is presented. The system consists of two machines, and all parts visit the two machines in the same order. The approach consists of first selecting a small number of part types for simultaneous production, in suitable ratios, in order to balance the workloads among the machines while not violating interstage buffer limitations. Production is organized in batches consisting of two part types each. Batches are then sequenced with the aim of minimizing the total number of tool changing operations. Detailed scheduling of each batch can be accomplished with no machine idle time and with minimum work-in-process requirements. Some advantages of the approach as presented here include management effectiveness and simplicity, in addition to other operational benefits inherent in a dynamic and flexible approach to part type selection (such as a higher utilization of the machines, a smoother tool changing activity, etcetera).

1. INTRODUCTION

This research addresses some particular planning problems in flexible flow systems. In particular, the focus of the research is concerned with determining how to produce over time parts of many types, each with different production requirements, on two different machines. All parts are processed first on machine M_1 and then on machine M_2. The proposed approach accounts for optimal machine utilization, storage limitations, and tooling issues such as tool magazine capacity and change-over time minimization.

In particular, approaches are specified to first partition all of the part types into overlapping batches, each batch to be produced consecutively over time. This is called *part type selection* (*PTS*). Each operation of each part type requires certain cutting tools that need to be placed in the corresponding machine's tool magazine. The PTS approach selects *tool-compatible* part types, i.e., part types that can be machined together subject to tool magazine capacity.

Then, we also address the problem of *batch sequencing* (*BS*), i.e., finding the best sequence in which the overlapping batches should be processed. It usually takes some time to change tools in between finished and new batches of part types. We suggest a batch sequencing policy that minimizes the total change-over time, when different part types require different tool sets. More generally, our policy minimizes the total number of times parts types are reprocessed. Since changing more part types means in general changing more tools, a good bound to the overall change-over time is achieved in this way. The approach here can accommodate any tool-changing mechanism, either manual or automatic. Depending on each specific situation, different objectives for BS can however be considered.

The problem of specifying how the parts of each type in each batch should be scheduled optimally is called *input sequencing* (*IS*). Two major objectives of IS are to maximize system utilization (minimize idle time) and to minimize the size of the buffer required to hold the waiting parts. The problem of minimizing the makespan without violating the buffer constraint is NP-hard in its general form (Papadimitriou and Kanellakis [1980]). However, we focus on the problem of finding, among the sequences minimizing the makespan, one that also minimizes the maximum

level of in-process inventory. This problem can be solved in polynomial time if the batch consists of two part types only (Agnetis et al. [1991]). Panwalkar [1991] provides an algorithm to determine an input sequence into a two-machine flow system that minimizes makespan. Travel time between the two machines by a single transporter is considered. We do not address IS further in this paper.

There is a related literature on FMS part type selection. One general approach to select part types for concurrent production over time is called *batching* (Whitney and Gaul [1985], Hwang [1986], Rajagopalan [1986]). All part types are partitioned into disjoint batches and all parts in a particular batch are machined continuously until all requirements of all part types are finished. Another PTS approach is the *flexible* approach (Stecke and Kim [1988, 1989, 1991]). All part types in a particular batch are machined in certain relative ratios that balance machine workloads until the requirements of some part type are finished. Then the tools required for the finished part type may be taken out of the magazines and the tools required for the next part type(s) to be produced are loaded into the magazines, if new part types are selected to join the remaining part types.

In general, following a flexible approach to system operation is better with respect to system operation performance than batching (Stecke and Kim [1988]). However, there are situations in which a batching approach is better (Stecke [1989]).

The approach to optimal PTS and BS here is based on the flexible approach. Many compatible part types are selected subject to tool magazine capacity. Then, a subset of two of these part types are selected at a time for immediate processing. As previously mentioned, these two can be machined in suitable ratios that balance machine workloads, and sequenced so that both idle time and buffer requirements are minimized (see Agnetis et al. [1991]), until the requirements of one part type in the batch are finished. Then a new part type is selected to be processed with the remaining part type in new optimal ratios that balance machine workloads. Furthermore, by machining only two part types at a time, usually most or all pairs are feasible with respect to tool magazine capacity. This helps to simplify the PTS.

The advantages of our PTS algorithm include both optimal system operation as well as ease of operation. Our suggested approach is of course appropriate for some, but not all, FMS situations. In particular, either the processing time per operation (defined by several cutting tools) and/or the production requirements per part type need to be large enough to ensure that a system setup (changing the tools for a new part type) does not occur very often. This will usually not be a problem.

Our global approach is especially useful in a dynamic environment, with orders arriving over time. Periodically, PTS and BS can be solved to optimally order the production of all part types. Near the end of a production period, there will be only a few part types remaining, all largely requiring the current bottleneck machine. At this point, the recently arrived orders can supplement the remaining orders and global PTS and BS are performed again.

The plan of the paper is as follows. The global solution approach is described in Section 2. In Section 3, we detail the global approach, and provide efficient solution algorithms for both PTS and BS. In Section 4, we provide a detailed example. In Section 5, numerical evidence of the proposed approach is reported. Finally, we conclude in Section 6 with future research needs.

2. PROBLEM STATEMENT AND SOLUTION APPROACH

Consider the following NP-hard scheduling problem (Papadimitriou and Kanellakis [1980]):

> *Given a two-machine flow shop and a set J of part types to be processed, find a schedule of parts over time such that the completion time is minimum and the maximum level reached by the interstage buffer never exceeds a fixed capacity* $q > 0$.

A solution to the above problem in real-world situations requires the consideration of some practical aspects. When s different part types are to be processed in the same mix, each machine must be equipped with a tool set which is the union of the tool sets needed by all selected part types: the larger the s, the higher the chance is that such tool set exceeds the capacity of the tool

magazine of some machine. It follows that it can be convenient to process only a few part types at a time, when this is sufficient to attain a good system performance. In the following, we say that *two or more* part types are *tool-compatible* if the tool set required for executing the corresponding operations can coexist in the appropriate tool magazines.

We suggest a solution approach to the above problem that accounts also for these tooling issues. With this approach, production occurs as follows: a batch of only two part types with certain workload balancing mix ratios is in process at any time; when the requirements of one part type are finished, another part type is selected to be processed with the type remaining in new mix ratios, and together they form a new batch.

Let the set P of all part types having production requirements be partitioned into two subsets, P_1 and P_2, so that if part type $i \in P_1$, (respectively, if $k \in P_2$) then a part of type i (of type k) requires a higher processing time on M_1 than on M_2 (on M_2 than on M_1). Since we look for a workload-balanced schedule of parts in each batch, one of the selected part types (say i) must belong to P_1 and the other (say k) to P_2. The detailed sequencing of the parts in the batch is performed according to the following rule:

Rule 1. *If a part of type $i \in P_1$ can be scheduled on M_1 without creating idle time on M_2, then schedule a part of type i.*

The following theorem holds:

Theorem 1. *If the part types in a particular batch are in workload-balancing mix ratios, then a schedule obtained by applying Rule 1 minimizes the maximum buffer level while keeping both machines always busy.*

Proof: See Agnetis et al. [1991].

In the following, let p_{ij} (p_{kj}) denote the *processing time* of a type i (k) part on M_j, j=1 or 2. The workload-balancing mix ratios can easily be computed a priori. Let ρ_{ik} denote the *number*

of parts of type k that must be processed in the batch per *one* part of type i in order to balance the workloads of M_1 and M_2. In general, ρ_{ik} is not integer, may attain any rational value greater than zero, and depends only on the processing times of a part of each type i and k on the two machines. These values are computed in Agnetis et al. (1991), where it is shown that

$$\rho_{ik} = (p_{k2} - p_{k1}) / (p_{i1} - p_{i2}). \quad (1)$$

The value of the *maximum buffer level*, q_{ik}, reached by scheduling parts of type i and k according to Rule 1 is also a function of the processing times. A table providing formulas for the computation of q_{ik} is given in Agnetis et al. [1991]. Alternatively, q_{ik} can be obtained by a simple simulation of the application of Rule 1.

The following example demonstrates the application of Rule 1.

Example 1. Consider two parts types 1 and 2, with processing times of $p_{11} = 7, p_{12} = 4, p_{21} = 5$, and $p_{22} = 12$ minutes. Therefore, part type 1 (part type 2) belongs to P_1 (P_2). The ratio ρ_{12} is given by $(p_{22} - p_{21} / p_{11} - p_{12}) = 7/3 \approx 2.33$. Figure 1 provides the schedule obtained by applying Rule 1 to a batch consisting of these two part types. Here we have sufficiently high requirements for both part types, and these are in the relative ratio ρ_{12}. The first part of the schedule is a part of type 2; after that, two parts of type 1 can be scheduled without creating idle time on M_2. Thereafter, the residual time on M_2 is 6 minutes: there is no room for another part of type 1, since $p_{11} = 7 > 6$. Therefore, a part of type 2 must be scheduled, followed by two parts of type 1. At this point, the residual time on M_2 is 7. Initially, it seems that there is room for another part of type 1, but notice that if a part of type 1 is scheduled, then the residual time on M_2 becomes 4, and this would create idle time on M_2, whichever the following part is. For this reason, a part of type 2 is scheduled, followed by three parts of type 1. At this point (Y in Figure 1), notice that the situation is exactly the same as that at the beginning (point X). In fact the total workload of both machines to point Y is $64 = 3p_{21} + 7p_{11} = 3p_{22} + 7p_{12}$. After point Y, the schedule obtained by following Rule 1 repeats identically over time. The cyclic input sequence of the two part types is: 211 211 2111.

By observing the schedule in Figure 1, we see that the buffer level never exceeds 1, i.e., in this example, $q_{12} = 1$.

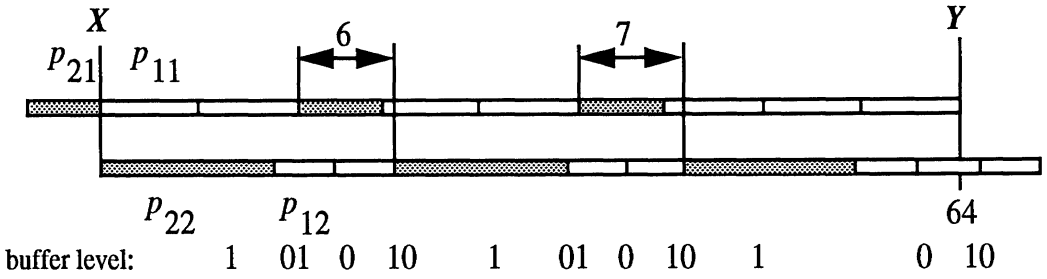


FIGURE 1. Schedule of Parts of Types 1 and 2 Following Rule 1.

It is not necessary to consider part types having the same processing times on the two machines. In this case, these parts can be scheduled consecutively, indeed, in any manner (up to the fulfilling of the requirements), realizing a perfect workload balance with no occupation of the buffer.

Clearly, the problem data may not always guarantee the existence of a solution such that M_2 is always busy. Thus, we actually analyze the problem of maximizing the length of the interval during which both machines are busy and the buffer capacity is not exceeded.

We say that two part types i and k are *workload-compatible* if and only if $i \in P_1$ and $k \in P_2$. A pair (i,k) of *tool-* and *workload-compatible* part types such that $q_{ik} \leq q$ is called *feasible.*

The general approach to the problems of PTS and BS may be summarized as follows.

PTS AND BS ALGORITHM

Step 1. Determine all of the tool- and workload-compatible pairs of part types (i,k), $i \in P_1$ and $k \in P_2$.

Step 2. For each such pair, compute the mix ratio ρ_{ik} and the buffer capacity required q_{ik}. Eliminate those pairs requiring a buffer that exceeds the actual system capacity q.

Step 3. *Part Type Selection*: Select a *set of pairs* of part types (*batches*), and for each pair, specify the number of parts to be produced (batch size), in order to maximize the interval in which both machines are busy.

Step 4. *Batch Sequencing*: Determine the order in which batches have to be processed.

Step 5. *Input Sequencing*: Sequence the parts of each batch according to the mix ratios computed at *Step 2*.

The remainder of the paper is concerned with the problems addressed in *Steps 3* and *4*, namely, *Part Type Selection* and *Batch Sequencing*. For the solutions of the problems defined in *Steps 2* and *5*, see Agnetis et al. [1991].

3. THE PART TYPE SELECTION APPROACH

The *Part Type Selection* (PTS) Problem can be defined as follows:

> *Given the requirements of each part type, determine which overlapping pairs of part types must be selected to form a batch, as well as how many parts, out of the requirements, must be produced for each part type in the batch, so as to maximize the interval in which both machines are busy without exceeding the buffer capacity.*

Let q denote the actual buffer capacity of the system. As before, q_{ik} denotes the *minimum buffer* capacity required to process parts of types i and k in the same batch with no idle time on either machine. Also, ρ_{ik} denotes the *number of parts* of type k that must be processed in the batch per *one* part of type i in order to balance the workloads of M_1 and M_2. From Theorem 1, recall that as long as parts are processed in the relative ratio ρ_{ik}, the buffer never exceeds q_{ik} and both machines are always busy.

Let us now introduce a bipartite graph $G = (U,V,E)$, where the vertex set U (V) is in a one-to-one correspondence with P_1 (P_2), and the edge set E is defined as follows:

$E \equiv \{(i,k) \mid pair\ (i,k)\ is\ feasible\}$.

Clearly, only pairs of part types belonging to E may be processed in the same batch.

In the remainder, we use the following notation:

- x_{ik} indicates the *number of parts of type* i that will be produced with *parts* of type k *in the same batch.* According to the previous definitions, the corresponding number of parts of type k is given by $\rho_{ik} x_{ik}$.
- n_i (n_k) denotes the requirements of part type i (part type k).

A formulation of the PTS Problem is the following, Problem (P1):

$$\textit{Problem (P1)} \qquad \max \sum_{(i,k)\in E} (p_{i1} + \rho_{ik} p_{k1}) x_{ik}$$

$$\text{s.t.} \qquad n_i' = \sum_{k:(i,k)\in E} x_{ik} \le n_i \qquad i \in P_1$$

$$n_k' = \sum_{i:(i,k)\in E} \rho_{ik} x_{ik} \le n_i \qquad k \in P_2$$

The first (second) set of constraints ensures that the number of parts i (parts k) to be produced does not exceed the requirements for that part type. The objective is to maximize the length of the interval during which both M_1 and M_2 are busy, i.e., the total parallelized workload. Since workload balance is achieved by sequencing the parts in the given mix ratios, it is sufficient to maximize the workload of M_1.

Obviously, the batch size (x_{ik}, $\rho_{ik} x_{ik}$) obtained from the solution of the linear Problem (P1) is generally not integer, even if x_{ik} is constrained to integrality, since in general ρ_{ik} is in turn noninteger. Therefore, a suitable rounding procedure is required. It turns out that a batch size obtained by rounding the solution of Problem (P1) is usually comparable, in terms of optimality, to a batch size obtained by rounding an integer solution of the same problem (see Section 5). This is especially true in the case of medium- or high-volume production, where, in our experience and for our purposes, rounding errors are usually negligible (see Arbib et al. [1991] and Stecke and Kim [1989]).

In general, a solution of Problem (P1) does not complete the production requirements n_i and n_k, for $i \in P_1$ and $k \in P_2$: only $n_i' \leq n_i$ ($n_k' \leq n_k$) parts of type $i \in P_1$ ($k \in P_2$) are scheduled. Therefore, in general one has to decide how to process the remaining parts. However, this is not a problem in a dynamic environment, in which new orders arrive to join the remaining ones to provide new problem input. In a static environment, the solution of Problem (P1) does not cover the production of some parts, and hence we must specify how to schedule such parts. The problem consists of assigning the residual leftover parts to batches, where the cost of an assignment is related to the machine idle time introduced. An assignment of residual parts of class P_1 (of class P_2) may be realized by appending the residual parts at the end (at the beginning) of the batch. It can be proved that be doing so, irregardless of the assignment, the completion time of each batch is still optimal, and the buffer required is not greater than that required when parts i and k are processed in the ratio ρ_{ik} (see Agnetis et al. [1991]).

An intermediate approach may be useful when the arrival rate of new orders is not large enough to allow the dynamic generation of a meaningful sequence of PTS problems. In this case, we can extend the set of feasible pairs by considering pairs (i,k) of part types such that $q_{ik} > q$. Sequencing the parts of such batches without exceeding the buffer capacity clearly implies that there will be a certain amount of idle time on one machine. This idle time can however be easily taken into account by slightly modifying the problem formulation. In particular, we may require that the sum of the idle times accumulated inside each batch does not exceed a given percentage W of the total parallelized workload, i.e., of the value of the objective function. Thus, we can add the following constraint to Problem (P1):

$$\sum_{(i,k)\in E} c_{ik}\, x_{ik} \leq W \sum_{(i,k)\in E} (p_{i1} + \rho_{ik}\, p_{j1}) x_{ik} \qquad (2)$$

where c_{ik} denotes the amount of idle time introduced in a batch (i,k) of size $(1,\rho_{ik})$.

In the following subsection, we analyze a structural property of basic feasible solutions of Problem (P1) and show how to exploit it for solving the Batch Sequencing problem.

3.1 A Property of the Basic Solutions to Problem (P1)

In this section, a structural property of the solutions to Problem (P1) that allows the *minimization of the total number of tool changes* is proved. This property results in more effective strategies for what concerns batch sequencing, as explained in Section 3.2.

In a feasible solution to Problem (P1), the values x_{ik} specify the numbers of parts of type i that must be produced paired with $\rho_{ik}\, x_{ik}$ parts of type k. Given a feasible solution $\{x\}$, let $G(x) = (U,V,E(x))$ be an *undirected* graph, where the edge set $E(x)$ is defined as follows:

$$E \equiv \{(i,k) \mid i \in P_1, k \in P_2, \text{ and } x_{ik} > 0\}.$$

Notice that Problem (P1) consists of n constraints, and therefore no more than n variables can be greater than zero in a basic feasible solution (bfs).

Theorem 2. If $\{x\}$ is a basic feasible solution to Problem (P1), then the graph $G(x)$ is acyclic.

Proof: Suppose that $\{x\}$ is a bfs, and that in $G(x)$ there is a cycle consisting of $2r$ edges. (Since $G(x)$ is bipartite, this cycle must be even). Starting from a part type belonging to P_1, let the part

1	0	0	...	0	1	0	0	...	0
0	1	0	...	0	0	1	0	...	0
0	0	1	...	0	0	0	1	...	0
0	0	0	...	1	0	0	0	...	1
0	$\frac{p_{r+1,2}-p_{r+1,1}}{p_{21}-p_{22}}$	0	...	0	$\frac{p_{r+1,2}-p_{r+1,1}}{p_{11}-p_{12}}$	0	0	...	0
0	0	$\frac{p_{r+2,2}-p_{r+2,1}}{p_{31}-p_{32}}$	...	0	0	$\frac{p_{r+2,2}-p_{r+2,1}}{p_{21}-p_{22}}$	0	...	0
0	0	0	...	0	0	0	$\frac{p_{r+3,2}-p_{r+3,1}}{p_{31}-p_{32}}$	...	0
•				•					•
•				•					•
•				•					•
0	0	0	...	$\frac{p_{2r-1,2}-p_{2r-1,1}}{p_{r1}-p_{r2}}$	0	0	0	...	0
$\frac{p_{2r,2}-p_{2r,1}}{p_{11}-p_{12}}$	0	0	...	0	0	0	0	...	$\frac{p_{2r-2}-p_{2r-1}}{p_{r1}-p_{r2}}$

FIGURE 2. The Matrix A_C.

types encountered on the cycle be: 1, r+1, 2, r+2, ..., r, 2r, where part types 1, 2, ..., r belong to P_1 and types r+1, r+2, ..., 2r belong to P_2. Let C be the set of columns corresponding to the edges in the cycle, and let A_C be the 2r x 2r submatrix obtained by selecting the rows and columns corresponding to the nodes in the cycle. Each column of A_C has 1 in the i-th row and ρ_{ik} in the k-th row, and the other elements are zero (see Figure 2). According to the definition of ρ_{ik}, if we divide the i-th row by $(p_{i1} - p_{i2})$, and the k-th row by $(p_{r+k,2} - p_{r+k,1})$, and add these two rows, we obtain a null row. This shows that A_C is not of full rank, and since the only nonzero elements of C lie in A_C, $\{x\}$ is not a bfs. Q.E.D.

3.2 Part Type Selection Over Time (Batch Sequencing)

In this section, we consider the problem of finding a good way to sequence the batches selected according to the method proposed using Problem (P1) for PTS. This problem is called the *Batch Sequencing* (BS) Problem. The objectives of BS may vary depending on the specific situation. In particular, different part types may require different tool sets, and hence one wants to minimize the total number of unnecessary re-tooling operations. We therefore require that, whenever possible, once a part type is input, it is processed until all the parts of that type that have been decided to be produced in the current planning period are finished. In the following, the notation (i,k) is employed to indicate the batch consisting of part types $i \in P_1$ and $k \in P_2$. Let $\{x^*\}$ indicate an optimal basic solution to Problem (P1).

Theorem 2 indicates that $G(x^*)$ is a forest. From the viewpoint of minimizing the total number of re-tooling operations, a very favorable situation occurs when each connected component of $G(x^*)$ is a path. In this case, batches can be sequenced so that each part type and the corresponding tool set are loaded only once. Namely, if one of such paths is {1, 2, 3, ..., m-1, m}, we can process the overlapping batches as follows: we start with the batch (1,2); after n'_1 parts of type 1 are finished, we replace such part type by part type 3, and therefore process the batch (3,2), and so on, up to the last batch (m-1, m).

Observe that this property holds for more general graphs than paths. The above discussion applies also if, for each connected component T of $G(x^*)$, there exists a path π (called a *dominating path*) such that each node of T either belongs to π, or is a leaf of T. In De Simone et al. [1990], components with such a structure are called *caterpillars*. For example, if $G(x^*)$ is the graph of part types in Figure 3, a batch sequence with the desired characteristics is: (1,4), (2,4), (3,4), (5,4), (7,6), (8,6), (9,6), (9,10), (9,11).

In general, some part types may have to be loaded more than once. However, the leaves of the forest $G(x^*)$ still correspond to part types (and tools) that need to be loaded only once. Let us therefore consider the forest $F(x^*)$ obtained from $G(x^*)$ by dropping all of its leaves. Let π denote

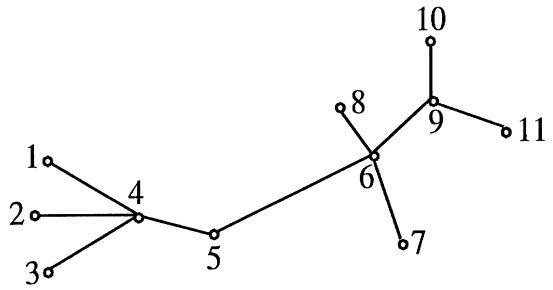


FIGURE 3. Graph $G(x^*)$ Having the Structure of a Caterpillar.

a path of part types on $F(x^*)$. From the above discussion, the general BS problem can be defined as follows:

Given an optimal solution x^* to Problem (P1), find a partition $\Pi \equiv \{\pi_1, ..., \pi_m\}$ of the forest $F(x^*)$ into a minimum number of edge-disjoint paths.

The number of paths in partition Π minus the number of connected components of $F(x^*)$ is equal to the total number of times that some part types are loaded more than once. To find an optimal partition Π, one can proceed as follows.

ALGORITHM TO OPTIMALLY PARTITION Π

Step 0. Set $\Pi \equiv \emptyset$.

Step 1. Find a path π linking two leaves of the forest $F(x^*)$.

Step 2. Add path π to partition Π and remove from $F(x^*)$ all edges of π.

Step 3. If $F(x^*)$ contains no edges, then stop; otherwise, go to *Step 1*.

It is easy to see that if ω denotes the number of odd-degree nodes of $F(x^*)$, then $|\Pi| = \omega/2$. The following Example 2 illustrates the BS approach.

Example 2. Consider the graph of part types, $G(x^*)$ of Figure 4a, obtained by solving Problem (P1). After removing its leaves, we obtain the graph $F(x^*)$ of Figure 4b. A partition $\Pi \equiv \{\{1,2,3,4\}, \{5,3,6\}, \{7,8,9\}\}$ of $F(x^*)$ into a minimum number of edge-disjoint paths is also shown. An optimal batch sequence of part types is (1',1), (1",1), (1"',1), (1,2), (**3**,2), (**3**,4), (4',4), (4",4), (5',5), (5",5), (**3**,5), (**3**,6), (6',6), (7,7'), (7,8), (8,8'), (8,9), (9',9). In this sequence, only part type 3 needs to be processed at two different times. In fact, we have $|\Pi| = 3$, and $F(x^*)$ has two connected components.

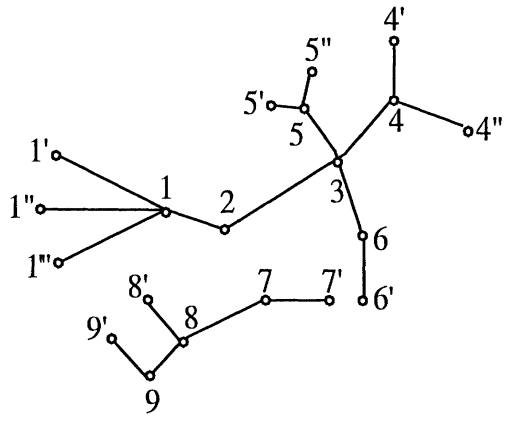


FIGURE 4a. A Forest $G(x^*)$ Obtained from the Solution x^* of Problem (P1).

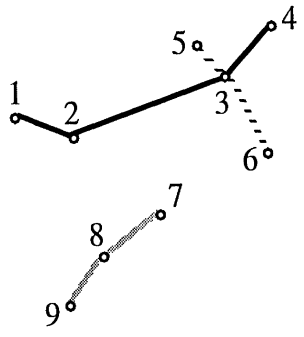


FIGURE 4b. The Forest $F(x^*)$ and a Partition of its Edges into a Minimum Number of Paths.

Finally, note that different objectives for the batch sequencing problem can be considered for different situations. Consider for example a situation in which, for any given batch, changing the tool set of one part type is as expensive or time-consuming as changing the tool sets of both. Clearly, a sequencing strategy such as the one described above may not be of interest. On the other hand, a possible objective is that of preventing buffer overflow during the transient period between any two consecutive batches, while also minimizing the total idle time. In this case, a feasible strategy is that of scheduling the first operation of the new batch on M_1 so that it ends exactly when the last operation of the previous batch is completed. The problem of computing a batch sequence that does not violate this constraint and minimizes the total idle time is a 2-machine No-Wait Flow Shop problem and can be solved in time $O(B \log B)$, where B denotes the number of batches (Gilmore and Gomory, 1963). Observe that this strategy can also accommodate re-tooling times, even if these depend on the particular pair of batches considered, as demonstrated in Figure 5.

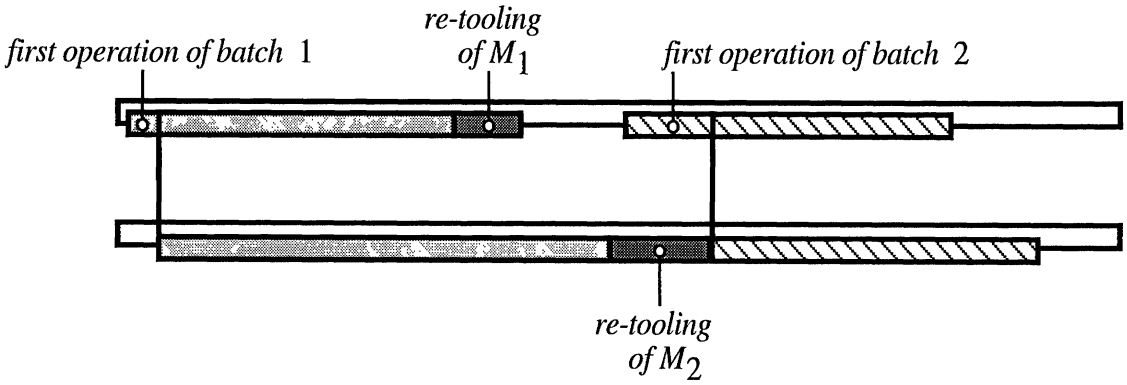


FIGURE 5. Transient Period Between Two Batches.

4. EXAMPLE

Now a detailed example is provided to show the application of the methodology described in Sections 2 and 3. Consider the production of 671 parts of ten different types, for a total workload amounting to 678 hours (about 28 days). The production of each part type requires a two-stage process. Production requirements, together with operation times (in seconds), and tool codes for each stage are indicated in Table 1 for each part type. In particular, part types 1, 3, 4, 5,

6, 7, 8, and 10 belong to P_1 whereas part types 2, 6, and 9 belong to P_2. Tools require either one slot in a tool magazine, or 3 slots if preceded by *. Table 2 summarizes total workload (hours) required per part type, as computed from the processing times of Table 1.

TABLE 1. Input Data for the Example Problem.

Part Type	*Production Requirements*	*Operation Time*	*Class*	*Tool Codes*
1	51	3013 1356	P_1	017 043 046 067 *072 076 077 079 083 087 101 106 *120 021 026 047 063 094 125 131
2	97	1502 2044	P_2	035 039 049 059 068 086 *096 113 *132 014 *036 037 050 051 *054 062 065 *078 *090 *108 124
3	64	2111 1647	P_1	008 010 013 016 022 029 *030 031 068 *084 109 002 008 033 051 082 095 104 117
4	59	1370 1286	P_1	*072 078 097 106 117 123 124 039 041 073 074 099 112 115
5	53	2524 1884	P_1	002 005 007 009 014 015 038 050 051 *054 *066 097 *132 *030 050 051 052 071 091 *120 *126 127
6	71	1456 1740	P_2	005 017 035 *048 086 093 103 118 009 017 051 063 069 079 081 112 116
7	64	1817 1502	P_1	011 047 057 074 076 082 *090 107 123 001 002 *006 040 *048 *054 055 056 081 098 112
8	54	2147 1588	P_1	005 010 014 *042 050 055 089 097 117 118 127 007 032 *042 051 098 101 103 121
9	98	1781 1866	P_2	001 013 *018 021 052 057 067 089 097 109 115 118 004 *006 011 033 045 069 070 116 121 129 131
10	60	2391 1673	P_1	001 016 040 050 063 067 080 087 088 092 101 116 008 011 015 *060 094 100 110

* Tools that require 3 slots.

The production system is a two-machine flexible flow line $\{M_1, M_2\}$ with a very limited intermediate buffer that can hold only one part at a time. Each machine is equipped with its own tool magazine. There are 132 different types of tools. From Table 1, each tool takes either 1 or 3 slots in a tool magazine. The larger-sized tools are about 15% of the total. The tool magazine capacity is 30 slots per machine.

TABLE 2. Total Workload (Hours) per Part Type and Operations.

Part Type	*Operation 1*	*Operation 2*	*Operations 1 and 2*
1	42.68	19.21	61.89
2	40.47	55.07	95.54
3	37.53	29.28	66.81
4	22.45	21.08	43.53
5	37.16	27.74	64.90
6	28.72	34.32	63.04
7	32.30	26.70	59.00
8	32.21	23.82	56.03
9	48.48	50.80	99.28
10	39.85	27.88	67.73
Total	361.85	315.90	677.75

The concurrent production of a pair (i and k) of part types ($i, k = 1,...,10, i \neq k$) is *feasible* if:

(i) Part types i and k are *workload-compatible*, i.e., $i \in P_1$ and $k \in P_2$;

(ii) Part types i and k are *tool-compatible*, i.e., the tool sets required for both i and k on M_1 and on M_2 take no more than 30 slots in each tool magazine;

(iii) It is possible to find a relative *part mix ratio* ρ_{ik} between i and k such that there exists a production sequence which does not create idle time on M_2 and never requires more than one part waiting in the intermediate buffer (q=1).

For all compatible pairs, the mix ratios ρ_{ik} can be obtained by equation (1) in Section 2 from the values of Table 1 and are listed in Table 3.

TABLE 3. Relative Production Ratios for Compatible Pairs of Part Types.

$\rho_{1,2} = 3.057$	$\rho_{1,6} = 5.835$	$\rho_{1,9} = 19.494$	$\rho_{3,2} = 0.856$
$\rho_{3,6} = 1.634$	$\rho_{3,9} = 5.459$	$\rho_{4,2} = 0.155$	$\rho_{4,6} = 0.296$
$\rho_{4,9} = 0.988$	$\rho_{5,6} = 2.254$	$\rho_{7,6} = 1.109$	$\rho_{7,9} = 3.706$
$\rho_{8,6} = 1.968$	$\rho_{8,9} = 6.576$	$\rho_{10,6} = 2.528$	$\rho_{10,9} = 8.447$

From the values of Tables 1 and 3, one can compute the maximum buffer requirement q_{ik} of each compatible pair, either from Table 1 in Agnetis et al. [1991], or by observation of the schedule, as done in Example 1 in Section 2.

The values of Table 1 also provide the tool magazine requirements of each compatible pair. For example, the concurrent production of part types 5 and 6 would require 20 tools (3 of which are of the large size) on M_1, for a global requirement of (3x3 + 17) = 26 slots, and 17 tools (3 of which are of the large size) on M_2, for (3x3 + 14) = 23 slots. Notice that tools 005 and 051 are required by both part types 5 and 6, on M_1 and M_2, respectively. Therefore, the pair (5,6) is feasible. On the other hand, the pair (2,5) is not tool-compatible, as the tools needed for the second operation require (3x7 + 12) = 33 slots, which exceeds the tool magazine capacity of M_2.

To define the feasible pairs of part types in this example, we use the bipartite graph $G = (U, V, E)$ of Figure 6. The white (black) nodes U (V) denote part types in P_1(in P_2), respectively. The edges of E denote the feasible pairs.

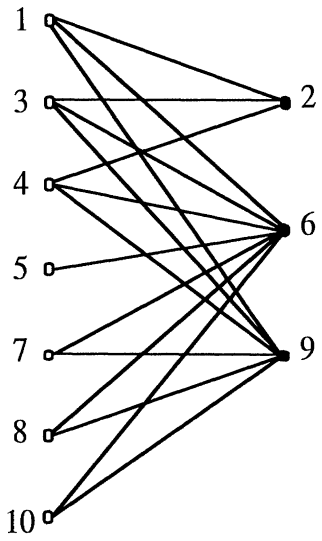


FIGURE 6. The Graph G of Feasible Pairs of Part Types.

The part type selection problem can be formulated as the integer linear program Problem (P1) of Section 3, as follows:

Example Problem (P1)

$$\text{maximize } 2.112x_{1,2} + 3.197x_{1,6} + 10.481x_{1,9} + 0.944x_{3,2} + 1.247x_{3,6} + 3.287x_{3,9} +$$
$$+ 0.445x_{4,2} + 0.500x_{4,5} + 0.869x_{4,9} + 1.612x_{5,6} + 0.953x_{7,6} +$$
$$+ 2.388x_{7,9} + 1.392x_{8,6} + 3.850x_{8,9} + 1.687x_{10,6} + 4.843x_{10,9}$$

subject to:

$$x_{1,2} + x_{1,6} + x_{1,9} \leq 51$$

$$x_{3,2} + x_{3,6} + x_{3,9} \leq 64$$

$$x_{4,2} + x_{4,6} + x_{4,9} \leq 59$$

$$x_{5,6} \leq 53$$

$$x_{7,6} + x_{7,9} \leq 64$$

$$x_{8,6} + x_{8,9} \leq 54$$

$$x_{10,6} + x_{10,9} \leq 64$$

$$3.057x_{1,2} + 0.856x_{3,2} + 0.155x_{4,2} \leq 97$$

$$5.835x_{1,6} + 1.634x_{3,6} + 0.296x_{4,6} + 2.254x_{5,6} + 1.109x_{7,6} + 1.968x_{8,6} + 2.528x_{10,6} \leq 71$$

$$19.494x_{1,9} + 5.459x_{3,9} + 0.988x_{4,9} + 3.706x_{7,9} + 6.576x_{8,9} + 8.447x_{10,9} \leq 98$$

$x_{ij} \geq 0$ and integer, $\forall (i,j) \in E$.

An optimum integer solution $\{x_{ik}^*\}$ to this integer program is given in Table 4, together with solutions $\{\xi_{ik}\}$ of the corresponding linear relaxation. The objective function values express the time interval (in hours) during which both M_1 and M_2 are busy. These values are very close, and can be compared to the bound given by the workload w of the least busy machine. (In this case, w = 315.9 hours. See Table 2.) This comparison clearly yields almost the same ratios for the two solutions, and allows us to conclude that more than 70% of the total workload can be assigned to the system with the described approach, for this example.

TABLE 4. Optimal Solutions to the Part Type Selection Problem.

$x_{1,2}^* = 11$	$x_{1,6}^* = 0$		$\xi_{1,2} = 10.8$	$\xi_{1,6} = 0.0$			
$x_{1,6}^* = 0$	$x_{3,2}^* = 64$		$\xi_{1,9} = 0.0$	$\xi_{3,2} = 64.0$			
$x_{3,6}^* = 0$	$x_{3,9}^* = 0$		$\xi_{3,6} = 0.0$	$\xi_{3,9} = 0.0$			
$x_{4,2}^* = 55$	$x_{4,6}^* = 0$		$\xi_{4,2} = 59.0$	$\xi_{4,6} = 0.0$			
$x_{4,9}^* = 4$	$x_{5,6}^* = 7$		$\xi_{4,9} = 0.0$	$\xi_{5,6} = 13.0$			
$x_{7,6}^* = 48$	$x_{7,9}^* = 16$		$\xi_{7,6} = 37.5$	$\xi_{7,9} = 26.4$			
$x_{8,6}^* = 1$	$x_{8,9}^* = 4$		$\xi_{8,6} = 0.0$	$\xi_{8,9} = 0.0$			
$x_{10,6}^* = 0$	$x_{10,9}^* = 1$		$\xi_{10,6} = 0.0$	$\xi_{10.9} = 0.0$			
Objective Function Value = 227.682			Objective Function Value = 228.135				
Objective Function Value/ Total Workload = .721			Objective Function Value/ Total Workload = .722				

The actual part type selection, of course, requires integer values not only for the number x_{ik} of parts of type i that must be processed in the same cycle with parts of type k, but also for the number $\rho_{ik}x_{ik}$ of parts of type k that vice-versa must be processed in the same cycle with parts of type i. A rounding procedure has therefore to be defined, not only for the solution $\{\xi_{ik}\}$ of the linear relaxation, but also for the integer optimum $\{x_{ik}^*\}$.

Figure 7 shows two square matrices which we shall refer to as *PTS matrices*, the entries of which correspond to pairs of part types. Entry (i,k) is equal to the actual number of parts of type i that must be processed in the same cycle with parts of type k. Such a number is defined for compatible pairs only, and the matrix is generally not symmetric.

A PTS matrix can be computed by applying a suitable rounding procedure to a feasible solution of Problem (P1). This rounding procedure should in general ensure that the result still represents a feasible solution of Problem (P1). Suppose that each component is rounded to the closest integer. Rounding down will never be a problem. On the other hand, among the candidate components, we may round up only those with the greatest fractional parts.

If we apply such a rounding procedure to the *relaxed* solution $\{\xi_{ik}\}$ of Example Problem (P1), we obtain the PTS matrix of Figure 7a. In particular, for each component $\xi_{ik} > 0$, entry (i,k) is obtained by rounding ξ_{ik}, and entry (k,i) is obtained by multiplying the rounded component by

the corresponding ρ_{ik}, and again rounding the result. Therefore, for example, in order to apply the suggested approach, 13 parts of type 5 have to be processed in the same cycle with 29 parts of type 6.

a.

	1	2	3	4	5	6	7	8	9	10	
1		11				0			0		11
2	34		55	8							97
3		64				0			0		64
4		59				0			0		59
5						**13**					13
6	0		0	0	**29**		42	0		0	71
7						38			26		64
8						0			0		0
9	0		0	0			96	0		0	96
10						0			0		0

b.

	1	2	3	4	5	6	7	8	9	10	
1		11				0			0		11
2	34		55	8							97
3		64				0			0		64
4		55				0			4		59
5						**7**					7
6	0		0	0	**16**		53	2		0	71
7						48			16		64
8						1			4		5
9	0		0	4			59	26		8	97
10						0			1		1

FIGURE 7. Actual Part Type Selections Obtained From: *(a)* The Linear Relaxation of Problem (P1), and *(b)* its Integer Optimum Solution.

Figure 7b shows the PTS matrix derived by applying the same rounding procedure to the *optimum* solution $\{x_{ik}^*\}$ of Example Problem (P1). In this case, seven parts of type 5 are to be processed in the same cycle with 16 parts of type 6. Each matrix is associated with a *PTS vector* indicating the total number of parts of each type to be processed with our approach.

Notice that the integer solution obtained by rounding $\{\xi_{ik}\}$ has a value of 227.284 hours, whereas the value of the solution corresponding to the integer optimum is 227.218. Hence the integer solution is still slightly worse than the rounded linear solution. Notice that in the case considered here, the optimum value of the objective function of Problem (P1) is 227.682, whereas the linear relaxation yields an optimum value of 228.135 (see Table 2).

As observed in Section 3 (Theorem 2), the particular structure of the linear program (P1) is such that the selected feasible pairs (i,k), i.e., those corresponding to $\xi_{ik} > 0$, define a forest on the bipartite graph G. In our example, such a subgraph has the particular caterpillar topology shown in Figure 8.

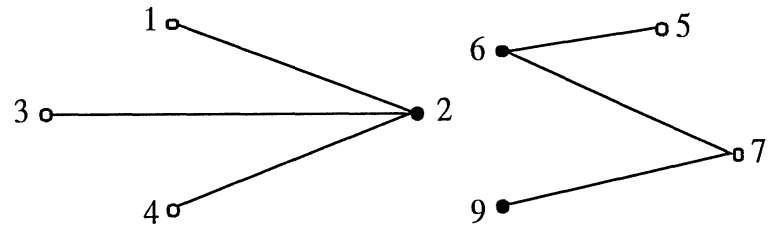


FIGURE 8. The Acyclic Subgraph of G Corresponding to the Feasible Pairs Selected Through the Linear Relaxation of the Example Problem (P1).

As discussed in Section 3.2, this topology can be used to determine an effective way of sequencing the different but overlapping cycles of part types belonging to the same connected component. For example, from Figure 8 we can begin with the coupled production of part types 5 and 6, then pass to part types 6 and 7, and finally to types 7 and 9. Similarly, one can process the other connected component, beginning with the production of part types 1 and 2, proceeding with the production of 3 and 2, and then with production of 4 and 2. By doing so, every part type is processed only once and totally in the same planning period, and therefore it is not necessary to load the corresponding tool sets more than once.

Observe that the described property does not hold in general for a nonbasic solution, and in particular for the integer optimum solution of Problem (P1). In the example presented here, the subgraph of G defined by those pairs (i,k) such that $x_{ik}^{*} > 0$ is given in Figure 9, and is exactly one cycle.

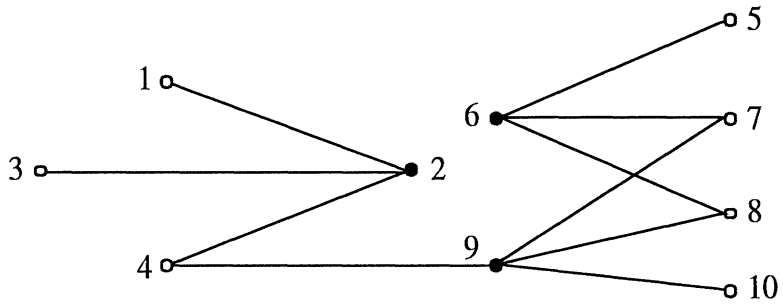


FIGURE 9. The Subgraph of *G* Defined by the Optimum Integer Solution of the Example Program (P1).

It can be seen from Figure 9 that it can happen that sequencing the various cycles may require the processing of at least one part type more than once.

5. COMPUTATIONAL RESULTS

In this section, we provide a vast collection of numerical results related to the part type selection and batch scheduling approaches described in Sections 2 and 3. The following analyses are the results of more than 600 sample problems of different sizes, generated with different distributions of operation times and tool requirements. All optimal solutions were found via the linear programming facility offered by the mixed-integer linear solver IBM MPSX-MIP/370, running on an IBM 3090 at the Università di Roma "La Sapienza". In all cases, the computation required only a few milliseconds of CPU time.

As it occurs in a mid-term planning situation, the problem instances generated are characterized typically by an average system workload not greater than 1 month. The required tools used are from 132 different cutter types. The capacity of the tool magazines of both machines of the system is 30 slots.

5.1 Quality of the Optimal Solutions of (P1) and Batch Sequencing Results

The first set of 80 test problems has been generated to produce 5, 10, 15, and 20 different part types in a flexible flow system with two machines and an intermediate buffer capacity of 1 or 2. For each part type, the system has to produce from 50 to 100 parts. The production of each part requires from 7 to 13 different tools, and from 7 to 16 different elementary operations, some of which may require the same tool. An elementary operation may take from 60 to 240 seconds. All data have been randomly generated with uniform probability distributions in the respective ranges.

The aim of this first test was on one hand to quantify the maximum system workload that can be totally processed in parallel with our approach, and on the other hand, to verify the possibility of sequencing all of the selected pairs without processing the same part type in more

than two different periods. Recall that this last possibility holds if all of the connected components of $G(x^*)$ have a caterpillar structure.

The results reported in Table 5 allow a comparison between the average optimum values z^* of the objective function of the linear relaxation of Problem (P1), and the average workload w of the least busy machine. Each value from each Test is obtained from 10 test problems of the same type. As observed, the value of w gives a trivial upper bound to the maximum period of parallel production. The last column of the table gives therefore an upper bound to the percentage of workload that is not parallelized with the proposed approach.

TABLE 5. Test Set I. Average Values of the Optimized Parallel Workload Compared with the Theoretical Bounds.

Test	*Number of Part Types*	*Buffer Capacity*	*w (Hours)*	*z* (Hours)*	*(w - z*)/w (%)*
1	5	1	162.5	133.7	17.7
2	5	2	152.7	130.3	14.7
3	10	1	346.6	277.8	19.8
4	10	2	330.8	279.3	15.6
5	15	1	499.4	454.1	9.1
6	15	2	510.1	456.2	10.6
7	20	1	687.1	621.8	9.5
8	20	2	706.5	646.1	8.5

With only one exception (Tests 5 and 6), the percentage of non-parallelized workload decreases as the buffer capacity increases. In general, these percentages have been under 20% of the upper bound of the theoretical parallelizable workload.

Table 6 gives, for the same Test Set, the average percentage of nodes that induce caterpillar components in the forest $G(x^*)$. From these values one can infer that should any two selected pairs require different tool sets on both machines (this property held for all of the problem instances generated in this test), then a high percentage of part types (from 87 to 100%) would require loading the corresponding tool sets only once. That is, all part types can be finished with

no re-tooling, therefore perfectly realizing the desirable operating mode corresponding to the flexible approach.

TABLE 6. Test Set I. Average Number of Part Types that can be Finished with No Re-tooling.

Test	*Number of Part Types*	*Buffer Capacity*	*Caterpillar Structure (% of Number of Part Types)*
1	5	1	100
2	5	2	100
3	10	1	99
4	10	2	98
5	15	1	93
6	15	2	96
7	20	1	92
8	20	2	87

In practice, all generated cases required a very low total number of tool-changing operations. The number of such operations tends to increase with the number of part types: as more nodes (part types) of G are available, chances decrease that the resulting solution forest only consists of caterpillars. The worst case observed had 20 nodes and required 4 tool changing operations over the theoretical bound that corresponded to a caterpillar structure of the solution forest.

5.2 Behavior of the Model with Unbalanced Workloads

A second Test Set of 458 problems was generated, corresponding to production of 15 or 20 different part types and buffer capacities of 1, 2, and 4. The objective of this test was to investigate the model behavior in cases of potentially unbalanced workloads or strong buffer requirements. Such situations may occur when the numbers of part types in the two different classes, as well as their operation times, differ significantly from each other.

There are three different sets of tests. The first set was intended to examine situations in which the processing times $p_{i2},p_{i1} << p_{k1},p_{k2}$ (or $p_{k1},p_{k2} << p_{i2},p_{i1}$), respectively, and has been organized as follows:

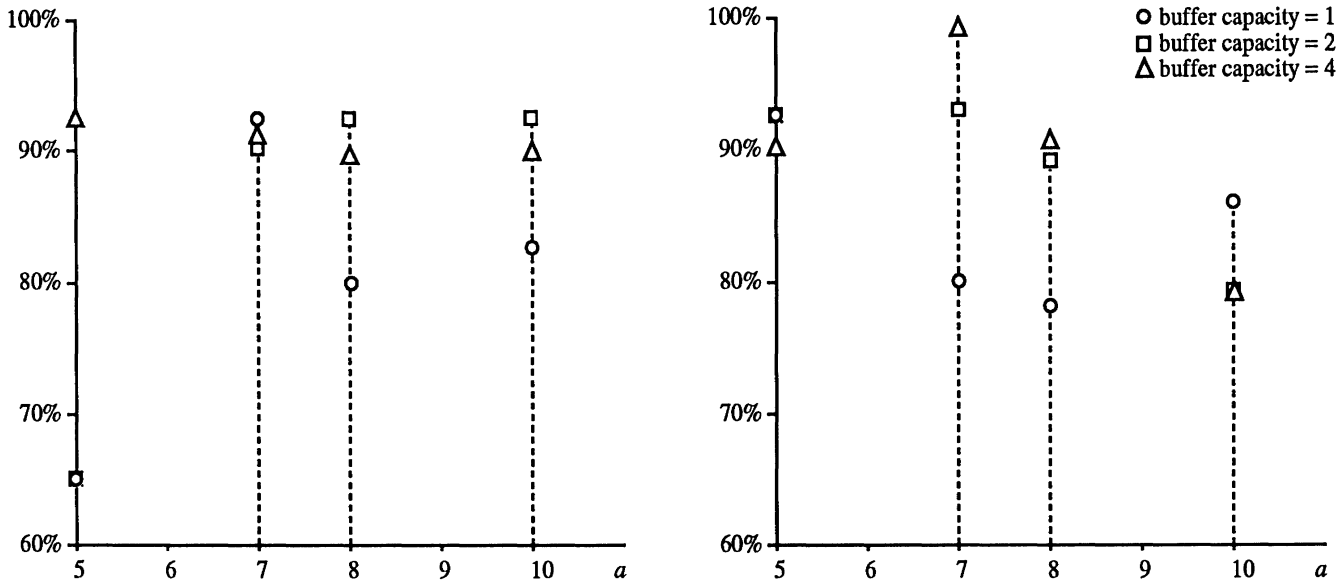
- Each problem instance has a part types of class P_1 and b part types of class P_2 ($a + b =$ 15 or 20).
- The operation times of part types of class P_1 (class P_1) are chosen randomly with uniform probability inside the interval $[a_1, a_2]$ (interval $[b_1, b_2]$). The times are in seconds.

For problems with 15 part types, we chose a = 5, 7, 8, and 10 and, respectively, b = 10, 8, 7, and 5. For each pair (a, b) such that $a + b$ = 15, two instances were generated with buffer capacities of 1, 2, and 4. Also, $[a_1, a_2]$ = [300, 1800] and $[b_1, b_2]$ = [1500, 3000], or vice-versa, $[a_1, a_2]$ = [1500, 3000] and $[b_1, b_2]$ = [300, 1800]. The total number of instances generated has therefore been (4 x 2 x 3 x 2) = 48.

For problems with 20 part types, we chose a = 5, 8, 10, 12, and 15 and, respectively, b = 15, 12, 10, 8, and 5. For each pair (a, b) such that $a + b$ = 20, we generated ten instances with buffer capacities of 1, 2, and 4. Also, $[a_1, a_2]$ = [300, 1800] and $[b_1, b_2]$ = [1500, 3000], or vice-versa, $[a_1, a_2]$ = [1500, 3000] and $[b_1, b_2]$ = [300, 1800]. The total number of instances generated in this case has been (5 x 10 x 3 x 2) = 300.

The experimental results obtained are provided in Figures 10 and 11. As a general feature (present in almost all generated cases), we observe a reasonable performance degradation for extreme (small and large) values of parameter a. This is expected in such unbalanced situations. As expected, the ratio between the optimized parallel workload and the theoretical bound generally improves for larger buffer capacities.

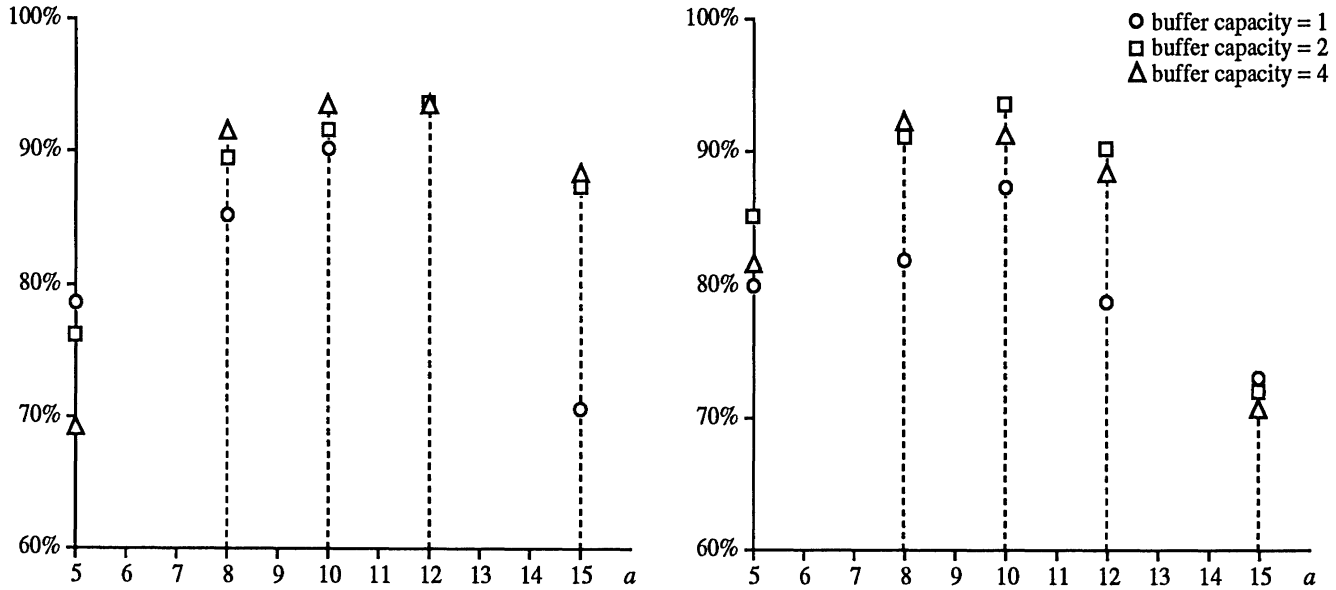
The second Test Set was based on the consideration that a critical situation when $\rho_{ik} > 1$ (respectively, when $\rho_{ik} < 1$) is one in which p_{k2} (respectively, p_{i1}) is much greater than the other operation times. In such cases, we have, respectively, $(p_{i1} - p_{i2}) << p_{k2}$ and $p_{k2} - p_{k1} << p_{i1}$, and the application of Rule 1 results in a large buffer requirement. Only a small number of feasible pairs can then be considered.



(a) $[a_1, a_2] = [300, 1800]$; $[b_1, b_2] = [1500, 3000]$.

(b) $[a_1, a_2] = [1500, 3000]$; $[b_1, b_2] = [300, 1800]$.

FIGURE 10. Test Set II. Average Values of the Ratio Between the Optimized Parallel Workload and the Theoretical Bound, for Problem Instances with 15 Part Types.



(a) $[a_1, a_2] = [300, 1800]$; $[b_1, b_2] = [1500, 3000]$.

(b) $[a_1, a_2] = [1500, 3000]$; $[b_1, b_2] = [300, 1800]$.

FIGURE 11. Test Set II. Average Values of the Ratio Between the Optimized Parallel Workload and the Theoretical Bound, for Problem Instances with 20 Part Types.

For example, suppose that we have 20 part types and the same values of a and b of the previous test. Let p_{i1}, p_{i2}, p_{k1} (or, respectively, p_{k1}, p_{k2}, p_{i2}) range between 300 and 900 seconds, and p_{k2} (or p_{i1}) range between 2,400 and 3,000. One can see that when the buffer capacity is 1 or 2, the values of q_{ik} obtained result in no feasible pair. We therefore set up this Test Set by generating 10 problem instances with the above ranges for the operation times and a buffer capacity of 4, for a total number of (5 x 10 x 2) = 100 instances. See Figure 12. Furthermore, when $a = 5$, $b = 15$ ($a = 15$, $b = 5$), and the expected value of p_{k2} (p_{i1}) is much greater than the other operation time, we obtain very unbalanced situations in terms of workload. One can see this from the corresponding poor average values of z^*/w (30.8% and 32.5% of total workload parallelization, respectively) reported in Figure 12.

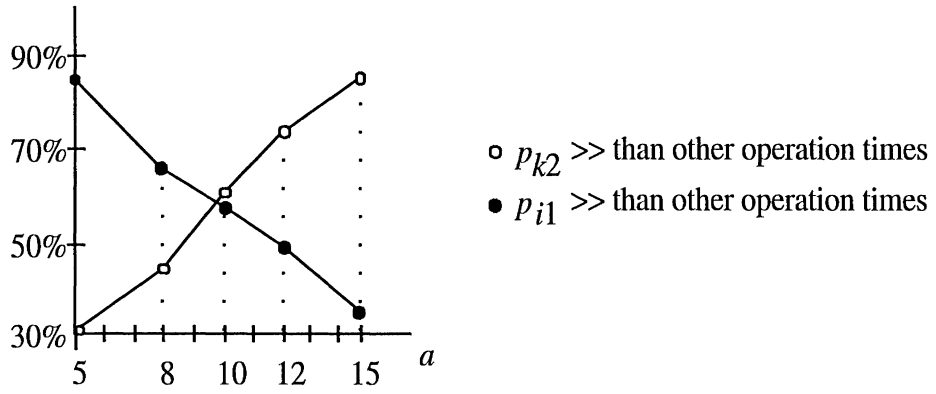


FIGURE 12. Test Set II. Average Values of the Ratio Between the Optimized Parallel Workload and the Theoretical Bound, for Problem Instances with 20 Part Types and Buffer Capacity = 4.

The next Test Set (on 10 problem instances with buffer capacity of 4 and 20 part types each) is considered in order to confirm the trend observed in the extreme situations. In this case, we set $a = 5$ and $b = 15$. The operation times of each process range as follows: p_{i1} and p_{i2} in [600, 900]; p_{k1} in [300, 600]; and p_{k2} in [2400, 3000]. On average, of 220.3 total hours of workload, our approach could be applied for 87.3 hours ($\approx$ 40%).

Each of the 458 problem instances generated in this set featured very good performance w.r.t. batch sequencing. About 88% to 100% of the part types can be machined without reloading

the corresponding tools sets. The worst case observed required only 3 re-tooling operations over the theoretical bound. However, this particularly good behavior stems from the fact that in unbalanced situations, the solution forest usually contains very few edges.

5.3 Bimodal Distributions of Times and Requirements

A third and last Test Set was set up in order to simulate a situation in which part types may be further partitioned according to short versus long operation times, and to small versus large production requirements. In particular, the total operation times of each part type range between 300 and 3,000 seconds, with a probability distribution having two maxima centered at 500-700 and 2,300-2,500 seconds. Similarly, the number of parts of each type required ranges between 30 and 120, with a probability distribution having maxima at 35-55 and 85-105 parts. (See Figure 13.) When generating each problem instance, we assume that operation times and production requirements are statistically independent variables. This means that to produce ten different part types, about two of them will on average have a total operation time less 12 minutes (≈ 700 seconds) and production requirements of at least 85 parts.

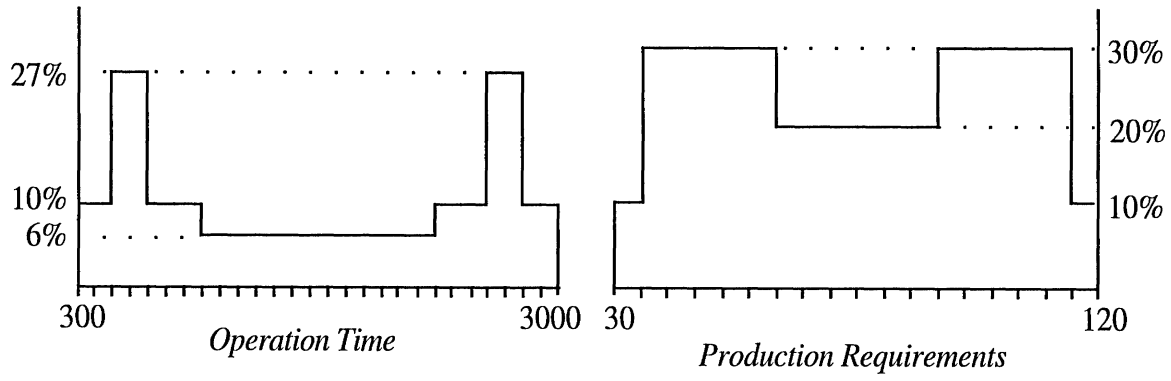


FIGURE 13. Test Set III. Probability Distributions (Percentage of Part Types) of Operation Times and Production Requirements.

With the above data, an appropriate range for the number of tools needed by each part type is 8 to 16. We generated 10 random problem instances with 5, 10, 15, and 20 part types each, with buffer capacities of 1 or 2, for a total number of 80 individual tests. The results are collected

in Tables 7 and 8. With respect to the results obtained with uniform probability distributions, the ratio $(w - z^*)/w$ shows a slight degradation. On the other hand, for each solution forest found, the average size of the caterpillar-structured subgraph appears slightly larger. However, as in the second Test Set, this behavior depends mainly on the low edge-density of the solution forests obtained.

TABLE 7. Test Set III. Average Values of the Optimized Parallel Workload Compared with the Theoretical Bounds.

Test	*Number of Part Types*	*Buffer Capacity*	*w (Hours)*	*z* (Hours)*	*(w - z*)/w (%)*
1	5	1	69.7	50.2	32.4
2	5	2	71.1	56.1	21.1
3	10	1	191.1	142.6	25.4
4	10	2	255.5	194.2	24.0
5	15	1	339.9	266.9	21.6
6	15	2	432.4	380.0	12.1
7	20	1	531.3	424.6	20.2
8	20	2	550.2	515.9	6.2

TABLE 8. Test Set III. Average Number of Part Types That Can Be Finished with No Re-tooling.

Test	*Number of Part Types*	*Buffer Capacity*	*Caterpillar Structure (% of Number of Part Types)*
1	5	1	100
2	5	2	100
3	10	1	100
4	10	2	99
5	15	1	100
6	15	2	100
7	20	1	96
8	20	2	88

We conclude by noting that for a few problems of small size (no more than ten part types), the *integer optimum* was found by means of an IBM MPSX-MIP/370 spending a considerable amount of computational effort. However, as noted in Section 3, no remarkable improvement in the quality of the solutions with respect to the linear relaxation has ever been noticed.

6. FUTURE RESEARCH NEEDS

This paper presents a new approach to part type selection in flexible flow lines, as well as to batch sequencing over time. To date, the analyses have concerned the two-machine with finite buffer situation. Among future problems that need to be addressed are the following:

1. Two-pools systems. Instead of two machines only, the system consists of two pools of m_1 and m_2 identical machines in each pool, respectively. The structure of the input sequencing problem is different. The solution will also depend on the number of buffer spaces and the location of the in-process inventories.
2. An m-machine flexible flow shop (with $m > 2$). For these larger types of systems, the number of part types to be selected for simultaneous production in order to guarantee workload balance will in general be greater than two. The input sequencing problem is likely to be much more complicated. The allowable numbers of buffers that can be between machines will also complicate the problem.

ACKNOWLEDGEMENTS

Kathy Stecke would like to acknowledge support from the Alexander von Humboldt Foundation in Bonn, the Università di Roma, and a summer research grant from the Business School of The University of Michigan. We also thank Marco Tassitano for providing the computational results.

REFERENCES

Agnetis, A., C. Arbib and K. E. Stecke, "Scheduling Two Part Types in a Two-machine Flexible Flow System", Working Paper, Università di Roma "La Sapienza", Dipartimento di Informatica e Sistemistica, Rome, 1991.

Arbib, C., M. Lucertini and F. Nicolò, "Optimal Workload Balance and Part Transfer in Flexible Manufacturing Systems", *International Journal of Flexible Manufacturing Systems*, Vol. 3, No. 1, pp. 5-25, February 1991.

Hwang, S., "Part Selection Problems in Flexible Manufacturing Systems Planning Stage", in: K. E. Stecke and R. Suri (eds.) *Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems: Operations Research Models and Applications*, Elsevier Science Publishers, B. V., Amsterdam, pp. 297-309, August 1986.

Panwalkar, S. S., "Scheduling for a Two-machine Flowshop with Travel Time Between Machines", *Journal of the Operational Research Society*, Vol. 42, No. 7, pp. 609-613, July 1991.

Papadimitriou, C. H. and P. C. Kanellakis, "Flow-shop Scheduling with Limited Temporary Storage", *Journal of the Association of Computing Machinery*, Vol. 27, pp. 533-549, 1980.

Rajagopalan, S., "Formulations and Heuristic Solutions for Parts Grouping and Tool Loading in Flexible Manufacturing Systems", in K. E. Stecke and R. Suri (eds.) *Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems: Operations Research Models and Applications*, Elsevier Science Publishers, B. V., Amsterdam, pp. 311-320, August 1986.

Stecke, K. E., "Procedures to Determine Part Mix Ratios in Flexible Manufacturing Systems", Working paper No. 587, The University of Michigan, School of Business Administration, Ann Arbor MI, October 1989.

Stecke, K. E. and I. Kim, "A Study of FMS Part Type Selection Approaches for Short-term Production Planning", *International Journal of Flexible Manufacturing Systems*, Vol. 1, No. 1, pp. 7-29, September 1988.

Stecke, K. E. and I. Kim, "Performance Evaluation for Systems of Pooled Machines of Unequal Sizes: Unbalancing Versus Balancing", *European Journal of Operational Research*, Vol. 42, pp. 22-38, 1989.

Stecke, K. E. and I. Kim, "A Flexible Approach to Part Type Selection in Flexible Flow Systems Using Part Mix Ratios", *International Journal of Production Research*, Vol. 29, No. 1, pp. 53-75, January-February, 1991.

Whitney, C. K. and T. S. Gaul, "Sequencial Decision Procedures for Batching and Balancing in FMSs", *Annals of Operations Research*, Vol. 3, pp. 301-316, 1985.