A PRODUCTION PLANNING PROBLEM IN FMS

*L. F. ESCUDERO; **G. PEREZ-SAINZ DE ROZAS

We present in this work a hierarchical approach for generating alternatives for production planning in a generic floor shop problem within the environment of Flexible Manufacturing Systems (hereafter, FMS). Briefly, the problem can be stated as follows: Given the resources of a FMS and the characteristics of the parts to be produced along a planning horizon, obtain the loading ordering of the parts in the FMS, the execution ordering of the operations and the processing route of each part (i.e., the working stations where each operation is to be executed), such that the production and transport costs are minimized and the modules workload is levelized. The problem is decomposed into three subproblems which are arranged in a hierarchy; a variety of models is presented, as well as the input/output relations that allow to integrate them; we also propose some algorithmic ideas to exploit the special structure of the problem. Computational experience is reported.

Keywords: Production planning, Integer applications

1. INTRODUCTION

The concept of FMS is largely accepted to be the foundation on which manufacturing companies should build the Factory of the Future; see Gunn (1982). A FMS basically consists of the following elements: A group of modules (i.e. working stations), such as numerically controlled machines, robots, etc. with automatic tool interchange processing capabilities; an automatic material handling system that links together the modules; and a computerized system that controls the production and transport systems.

Elsewhere (1986) we presents the framework for the models that could form a substantial part of a computerized system for production planning in FMS, such that it allows interactive model building, testing and experimentation. This paper deals with some of these models as one of the solution methods for a generic floor shop problem that frequently arises in FMS production planning (see Section 2). The main goal of the paper consists of presenting a hierarchical approach to this very complex problem (see Sections 3 to 5). As a secondary objective, we give the main ideas behind the algorithms that exploit the special structure of the problem. Computational results are reported for each model in the appropriate sections. Finally, Section 6 offers some conclusions and discusses some topics for future research.

2. PROBLEM DESCRIPTION

2.1. ELEMENTS

Let us describe the elements of the problem addressed in this work, and give the related notation.

• *L.F. Escudero; **G. Perez-Sainz de Rozas.
*IBM German Manufacturing Technology Center, -Max-Eyth Strasse, 6, Sindelfingen, F.R. Germany y Centro de Investigación UAM-IBM, Madrid.
**Facultad de Ciencias, Universidad del País Vasco.
• Article rebut el març de 1987.

A **process**, say f consists of the capability for any transformation/assembling of material requiring inputs of labor, machine time, other material and energy. Let F denote the set of processes that may be performed in a given FMS.

A **part type** is a set of identical parts to be produced on a given planning horizon. Let I denote the set of part types, and D_i the number of parts of type i for $i \in I$.

An **operation** consists of the technical requirements of a given process when performed on a part. Let N_i denote the set of operations that defines each part of type i for $i \in I$.

Let M denote the set of processing modules (hereafter, **modules**); and F_m, U_m, S_m and T_m the set of processing capabilities, capacity of the tools magazine, number of servers and available number of time units in module m for $m \in M$, respectively. Let R denote the set of resources required by the set F, and B_r and C_r the available amount and cost unit of resource r for $r \in R$, respectively.

The elements related to each process, say f for $f \in F$ are as follows: M_f, set of modules that have the processing capability f; u_f , number of tools that are required for performing the process; and $\bar{a}_{r,f}$ amount of the resource r for $r \in R$ required per time unit by the process f.

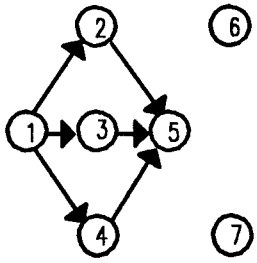
The elements related to each operation, say n for $n \in N_i$ for the part type i are as follows: $p(n)$, **associated process** for $p(n) \in F$; and $t_{i,n}$, **processing time** that is required by the operation n, provided that it is executed by using the most efficient module for performing $p(n)$ (see below).

Let the matrix E give the **efficiency** coefficients of the modules, such that $e_{m,f} = 1$ if m is the **most efficient module** for the execution of the process f and, otherwise, $0 < e_{m,f} < 1$ will give the related efficiency coefficient for $f \in F_m$ and $m \in M$; then, $t_{i,n}/e_{m,p(n)}$ gives the execution time of n while been performed by m.

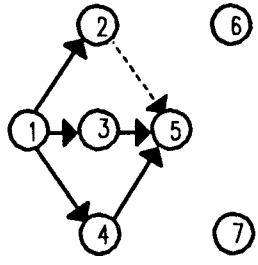
The availability of the segments of the **transport** system can be represented by the matrix Q, such that $q_{m1,m2} = \infty$ means that a part cannot be transported from the module $m1$ to the module $m2$; otherwise, it gives the inter-modules transport cost/time. Note that $q_{m,m} \neq \infty$ is allowed; it could be the case for which two consecutive operations are executed in the same module; usually, its value is zero. Haines (1985) gives a specialization of the shortest path algorithm for obtaining the matrix Q in a variety of material handling systems.

The precedence graph (or, more precisely, the graph of **precedence relationships** in the execution of the operations) of the part type i is formally denoted by the direct graph $G_i = (N_i, A_{i,1} \cup A_{i,2})$, where N_i gives the set of nodes (i.e., the set of operations), and $A_{i,1} \cup A_{i,2}$ represents the set of directed arcs; $(n1,n2) \in A_{i,1}$ means that the execution of the operations $n1$ and $n2$ has a **direct precedence** relationship (see Chatterjee et al., 1984) in the sense that, although it is admissible that other operations are executed in-between, the execution of $n1$ must be performed before the execution of $n2$; $(n1,n2) \in A_{i,2}$ means that $n1$ and $n2$ have an **immediate precedence** relationship (i.e., $n1$ must precede $n2$ and no other operation is allowed in-between). Case a, Figure 1 allows the sequence 1,2,3,4,5,6,7; it is not allowed by case b. Both cases allow e.g. the sequence 7,1,6,3,4,2,5; and no case allows e.g. 7,3,6,1,4,2,5.

The data related to the strategies for using the FMS along the planning horizon are as follows: $\overline{m}$, maximum number of modules where a processing capability may be assigned; $\underline{w}_m$, minimum workload per module's server that may be assigned to any process. Note: in addition to the above notation, each model requires its own; where some model requires the notation given for the others, it will be noted explicitly.



Case a: direct relationships



Case b: direct/immediate relationships

Case a: $A_{i,1} = \{(1,2),(1,3),(1,4),(2,5),(3,5),(4,5)\}$, and $A_{i,2} = \{\emptyset\}$
Case b: $A_{i,1} = \{(1,2),(1,3),(1,4),(3,5),(4,5)\}$, and $A_{i,2} = \{(2,5)\}$

Figure 1. Graphs of precedence relationships

2.2. PLANNING TASK

Let us consider the following assumptions:

1. Each operation to be performed on a given part can only be executed by one module, although a process can be assigned to more than one; i.e., it is allowed that the same operation be executed by different modules but only for different parts. Let $pp_{i,n,m}$ denote the **(production) proportion** of the part type i whose operation n will be performed by the module m.

2. The parts that belong to the same type will have the same **execution ordering** in the operations. Let $so_h|i$ denote the operation that will be executed at the h^{th} level for the part type i. Let $n1 \rightarrow n2|i$ denote the **partial ordering** $n1 = so_h|i$ and $n2 = so_{h+1}|i$. Let $rp_{m1,m2}|n1 \rightarrow n2|i$ denote the **(routing) proportion** of the parts of type i that will be routed from module $m1$ to module $m2$, where the operations $n1$ and $n2$ will be executed, respectively.

Finally, let J denote the whole set of parts to be processed along the planning horizon, such that

$$|J| = \sum_{i \in I} D_i \tag{2.1}$$

Given the assumptions, resources and part type characteristics described above, the goal consists of obtaining:

1. The **loading ordering** of the set J in the FMS; let $lo(j)$ denote the type of the part to be loaded at the j^{th} level for $j \in J$.

2. The execution ordering $\{so_h|i\}$ of the operations per each part type.

3. The **processing route** of each part along the FMS; let $pr_{j,n}$ denote the module where the operation n will be executed for $n \in N_{lo(j)}$ and $j \in J$.

such that some target is achieved.

Stecke (1983, 1986) and Abraham et al. (1985) suggest the targets that ideally should be achieved. In our case, the targets are: minimizing a production cost function (or, alternatively, levelizing the modules workload assignment), minimizing the transport cost/time, and balancing the modules workload along the time units of the planning horizon, such that the following constraints are satisfied:

1. The whole production volume is processed.
2. The amount to be used per resource can not exceed the maximum availability.
3. A process can not be assigned to more modules than a given maximum.
4. The capacity of the tools magazine of each module can not be exceeded.
5. The modules workload can not required more time units than the maximum available per module.
6. The modules workload per each process can not be less than a given minimum, provided that the process is assigned to the module.
7. The execution ordering of the operations per each part type must allow to execute all operations without violating the precedence relationships, nor the constraints related to the inter-modules transport system.
8. Each part is loaded in the FMS just only once, and its processing route assigns only one module to each operation.

2.3. A HIERARCHICAL APPROACH

The problem described above is a large-scale multicriteria mixed combinatorial problem; we can recognize resource allocation constraints in 1 to 6 above, structured combinatorial constraints as the **Asymmetric Traveling Salesman Problem** (hereafter, ATSP) with side constraints in 7, and combinatorial-like constraints in 8 whose structure has not yet been fully studied. We are not aware of any attempt to solve this generic floor shop problem. The next sections describe our (inexact) solution procedure; it uses a hierarchical approach and gives (hopefully, good) solutions without proving optimality even for each particular model. We believe that it is, at most, what the current algorithmic methodology can provide for helping to solve, in affordable time, this type of problems. The approach is as follows (see Figure 2):

1. Model **Modules workload assignment**. It obtains the processing capabilities to be used by each module, and its related modules workload assigned to each process, such that a production cost function is minimized. See Section 3. The output $\{pp_{i,n,m}\}$ is used by model 2.

2. Model **Operations execution sequencing and part type routing**. It obtains the operations execution sequence $\{so_n|i\}$ of each part type, and the related routing proportion $rp_{m1,m2}|n1 \rightarrow n2|i$ assigned to each inter-modules path, such that a transport cost/time function is minimized. See Section 4. The output is used by model 3.

3. Model **Parts loading sequencing and processing route in the FMS**. It obtains the type $lo(j)$ and the processing route $\{pr_{j,n}\}$ of the part that is to be loaded in the FMS at the level j for $j = 1,2,...,|J|$, such that the modules workload is balanced along the time units of the planning horizon. The output should be used by an **evaluative** system (see Engelke et al., 1983); it advances the performance of the proposed production planning, by using probabilistic information in a more de-aggregated environment (see Escudero, 1986).

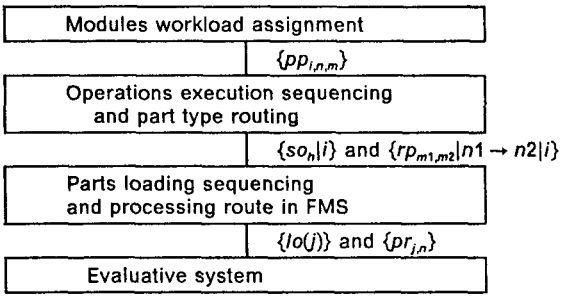


Figure 2. The hierarchical approach

3. MODULES WORKLOAD ASSIGNMENT

3.1. PLANNING TASK SUPPORTED BY THE MODEL

Assigning the processing capabilities to be used by each module along the planning horizon, and obtaining the related modules workload and the production proportions $\{pp_{i,n,m}\}$ of the part types. The assignment must be such that it allows to process the production volume programmed for each part type, the constraints stated below (see Section 3.2) are satisfied and the **resources cost** is minimized. Alternatively, any of the following targets could be achieved: Minimize the **total absolute deviation** of the modules workload from the mean workload, minimize the **greatest absolute deviation** of the modules workload from the mean workload, minimize the **greatest difference** on the modules workload, and minimize the **greatest** workload in any module; the goal in this case consists of balancing the modules workload and, as a result, getting an acceptable production rate and avoiding production 'bottlenecks'.

3.2. MODELIZATION

Let us state the additional notation required by the model's formulation. $l_{i,n,m}$ will denote the workload required by the operation n for $n \in N_i$, if it is to be executed by the module m for the whole part type i; it can be written

$$l_{i,n,m} = \frac{t_{i,n}}{e_{m,p(n)}} D_i \tag{3.1a}$$

Let $b_{i,n,m}$ denote the workload per server in module m related to the workload $l_{i,n,m}$, such that

$$b_{i,n,m} = \frac{l_{i,n,m}}{S_m} \tag{3.1b}$$

$a_{r,i,n,m}$ will denote the amount of resource r required by the workload $l_{i,n,m}$, such that

$$a_{r,i,n,m} = l_{i,n,m} \bar{a}_{r,p(n)} \tag{3.1c}$$

$c_{i,n,m}$ will denote the resource cost of the workload $l_{i,n,m}$, such that

$$c_{i,n,m} = \sum_{r \in R} C_r a_{r,i,n,m} \tag{3.1d}$$

Finally, let $y_{m,f}$ be a 0-1 variable such that it will take the value 1 if the processing capability f is (partially or totally) assigned to the module m and, otherwise, it is zero.

Assuming that the objective consists of minimizing the resource cost, the problem has the following formulation.

$$\min\ z = \sum_{i \in I} \sum_{n \in N_i} \sum_{m \in M_{p(n)}} c_{i,n,m} pp_{i,n,m} \tag{3.2a}$$

subject to

$$\sum_{i \in I} \sum_{n \in N_i} \sum_{m \in M_{p(n)}} a_{r,i,m,n} pp_{i,n,m} \leq B_r \quad \forall r \in R \tag{3.3a}$$

$$\sum_{i \in I} \sum_{n \in N_i | p(n) \in F_m} b_{i,n,m} pp_{i,n,m} \leq T_m \quad \forall m \in M \tag{3.4a}$$

$$\sum_{m \in M_{p(n)}} pp_{i,n,m} = 1 \quad \forall n \in N_i,\ i \in I \tag{3.5}$$

$$\underline{w}_m y_{m,f} \leq \sum_{i \in I} \sum_{n \in N_i | p(n) = f} b_{i,n,m} pp_{i,n,m} \leq \overline{w}_{m,f} y_{m,f} \quad \forall f \in F_m,\ m \in M \tag{3.6}$$

$$\underline{m}_f \leq \sum_{m \in M_f} y_{m,f} \leq \overline{m} \quad \forall f \in F \tag{3.7a}$$

$$\sum_{f \in F_m} u_f y_{m,f} \leq U_m \quad \forall m \in M \tag{3.8}$$

$$y_{m,f} \in \{0;1\} \quad \forall f \in F_m,\ m \in M \tag{3.9}$$

$$pp_{i,n,m} \geq 0 \quad \forall m \in M_{p(n)},\ n \in N_i,\ i \in I \tag{3.10}$$

Constraints (3.3) prevent to use more amount per resource than the maximum allowed. (3.4) avoid that the server's workload exceed the available time given to its module. (3.5) force to process the whole production volume of each part type. (3.6) state that the server's workload assigned to each process cannot be less than a given minimum, if any; $\overline{w}_{m,f}$ is a realistic upper bound, such that

$$\overline{w}_{m,f} \leq min\{\tilde{l}_{m,f},\ T_m\} \tag{3.11}$$

where $\tilde{l}_{m,f}$ gives the server's workload of the module m assigned to the process f, provided that it is the unique module where the process is assigned, such that

$$\tilde{l}_{m,f} = \sum_{i \in I} \sum_{n \in N_i | p(n) = f} b_{i,n,m} \quad \forall f \in F_m,\ m \in M \tag{3.12}$$

Note that $\overline{w}_{m,f}$ gives the 'big M' and, then, it must be kept as small as possible. (3.7) avoid that a process be assigned to more modules than the maximum allowed. The lower bound $\underline{m}_f$ is an strictly positive integer parameter that must be kept as big as possible; very frequently, it can be set to 2 or 3 without cutting off any feasible integer solution (see below); on the other hand, note that usually $\overline{m}$ is small (typical values, 3 and 4). (3.8) prevent that the number of tools to be used by each module be greater than the capacity of its tools magazine. Note that $pp_{i,n,m}$ is a 0-1 continuous variable; then, $pp_{i,n,m} D_i$ must be rounded to its nearest integer.

The model (3.2)-(3.10) is a large scale mixed 0-1 programming with a special structure in the constraints matrix that the algorithm should exploit; see below. Note that the problem's di-

mensions may be very high; in this case, it is necessary to sacrifice either the optimality's guarantee of the solution (by using an heuristic approach) or the model's accuracy (by solving an approximation of the problem). For dimensions within the range of 300 0-1 integer variables, 3000 continuous variables and 1000 constraints, a tight LP reformulation still allows to obtain q-sub optimal solutions in affordable time. In any case, q=2% seems to be quite adequate given the aggregation performed in the model's formulation.

Should the goal be minimizing the **greatest workload**, say $\bar{l}$ the model can be expressed

$$\min \{\bar{l}\} \tag{3.13a}$$

subject to (3.3)-(3.10) and

$$\sum_{i\in I} \sum_{n\in N_i | p(n)\in F_m} b_{i,n,m} pp_{i,n,m} \leq \bar{l} \quad \forall m \in M \tag{3.13b}$$

Note that $max\{T_m \ \forall m \in M\}$ gives the first upper bound of $\bar{l}$.

3.3. CASE $\overline{m} = 1$

The problem's dimensions are strongly reduced if $\overline{m} = 1$ (i.e., a processing capability cannot be assigned to more than one module). The problem consists of obtaining the module where each process is to be assigned, such that the resource cost, or alternatively the greatest workload, are minimized subject to the constraints stated below. Let the following additional notation. $\tilde{c}_{m,f}$ will give the cost of the resources required by the process f in the module m; it can be written

$$\tilde{c}_{m,f} = \sum_{i\in I} \sum_{n\in N_i | p(n)=f} c_{i,n,m} \tag{3.14}$$

$\tilde{a}_{r,m,f}$ will give the amount of the resource r to be used by the process f in the module m; it can be written

$$\tilde{a}_{r,m,f} = \sum_{i\in I} \sum_{n\in N_i | p(n)=f} a_{r,i,n,m} \tag{3.15}$$

The formulation of the model is as follows.

$$\min \ z = \sum_{f\in F_m} \sum_{m\in M} \tilde{c}_{m,f} y_{m,f} \tag{3.2b}$$

subject to the constraint (3.8) related to the capacity of the tools magazine, the 0-1 integrality constraint on $y_{m,f}$ (3.9) and

$$\sum_{f\in F_m} \sum_{m\in M} \tilde{a}_{r,m,f} y_{m,f} \leq B_r \quad \forall r \in R \tag{3.3b}$$

$$\sum_{f\in F_m} \tilde{l}_{m,f} y_{m,f} \leq T_m \quad \forall m \in M \tag{3.4b}$$

$$\sum_{m\in M_f} y_{m,f} = 1 \quad \forall f \in F \tag{3.7b}$$

Constraints (3.3) prevent to use more amount per resource than the maximum allowed. (3.4) avoid that the server's workload exceed the available time given to its module. (3.7) force just only one module per process.

The above model is a pure 0-1 programming that imbeds very well know structures that are exploited by the algorithm given in Section 3.4; the S1-entities (3.7) are explicitly used for variables fixing and branching selection in the branch-and-bound phase; and the coefficients reduction and constraints generation (by analyzing minimal (lifted) cover and k-configurations) are performed by using the constraints (3.3), (3.4) and (3.8). Note that the constraints (3.6) are not explicitly included in the above model; they are analyzed by the preprocessing phase such that $y_{m,f}$ is fixed to zero if it holds $\underline{w}_m > \tilde{l}_{m,f}$.

3.4. ALGORITHMIC APPROACH

The overall framework is given in Figure 3; it follows the description given in Crowder et al. (1983), Grotschel et al. (1984) and Hoffman and Padberg (1986) for pure 0-1 problems, but it is quite suitable for our purposes. The approach is valid for the models given in the two previous sections; where the structure of the model for $\overline{m} = 1$ can be exploited, it will be mentioned explicitly.

See other approaches in Stecke and Talbot (1983), Ammons et al. (1984), Chackravarty and Shtub (1984), Kusiak (1985), Whitney and Gaul (1984), Stecke and Moring (1985), Wittrock (1986) and Stecke (1986) among others. Typically, the resources based constraints (3.3) are not allowed, nor the $\underline{w}_m$-based constraints (3.5), $e_{m,f} = 1\ \forall f,m$, $\overline{m} = 1$ and some heuristics are used. See in Stecke (1983) the description of the model and in Berrada and Stecke (1983) the related algorithm for the case of tools sharing by different processes.

Let us give the main steps of our algorithm and, next, some insights:

1. We start reading the data from the data base.

2. The **preprocessing** phase inspects the data automatically and a permanent problem reduction may be carried out. Such problem reductions concern:

   a. Fixing the y-variables due to feasibility considerations to either zero or one (see Guignard and Spielberg 1977, Johnson et al. 1985, and Freville and Plateau 1986), such that they are carried out by performing independently feasibility tests on the data as well as by partially using the **probing** procedure (note that $y_{m,f} = 0$ implies $pp_{i,n,m} = 0$ for $\forall n \in N_i | p(n) = f,\ i \in I$).

   b. Coefficient reduction in some types of constraints; as a by-product it clears the constraints of common divisors greater than one and reduces the 'big M's'.

   c. Tightening the range of variability of some types of constraints.

   d. Deleting from the model redundant and in any case non-binding constraints.

   The preprocessing phase also analyses the implications of the previous reductions such that it could produce new reductions on the problem. After preprocessing, we have the definite version of the problem; let us name it the **global problem** to be optimized.

3. Ad-hoc heuristic algorithms are used to find the first upper bound of the optimal solution.

4. Since the value of $\overline{m}$ (see constraints (3.7)) is not too-high, a great number of variables will be zero in the optimal solution; then, one proceeds next with the selection of a subproblem for the optimization if $\overline{m} > 1$; let us name it the **quasi-global problem**. It will be included by the variables with non-zero value in the solution provided by the heuristic algorithm, if any.

5. Quasi-global problem set-up.

6. LP quasi-global problem optimization by relaxing the integrality constraint of the y-variables.

7. To prove that the optimal solution to the LP quasi-global problem is also optimal to the LP global problem, one needs to 'price out' the variables that were not included in the quasi-global problem (and, then, with zero value).

8. If the reduced cost of any of these variables has not the appropriate sign, the quasi-global problem is revised by including these variables, the current LP solution is restored and go to Step 6.

9. If the reduced cost of the above variables has the appropriate sign, we have obtained a lower bound on the optimal solution of the global problem. Now, if the reduced cost of the nonbasic y-variables exceed the gap between the lower bound and the best integer solution obtained so far (up to now, the heuristic solution if any) then the related variable will be permanently fixed to its current value; note that it could be either zero or one. If, as a result of the testing, all nonbasic y-variables have been fixed then stop since we have obtained the optimal solution of the global problem.

10. If the number of variables that have been fixed in the previous step is greater than a given bound then the preprocessing phase is used again but, now, without using probing. Then, go to Step 2.

11. Otherwise, the quasi-global problem is revised by deleting the fixed variables if any and, then, the LP current solution is restored.

12. Once it has been tested that no further problem reductions can be performed with the above procedures, we proceed to the **constraints generation** phase, by using the constraints where only the y-variables have non-zero coefficients (see below).

13. If the attempt is successful, the quasi-global problem is revised by plugging in the new constraints. Next, the current LP solution is restored and the LP optimization is reassumed from that basis by going to Step 6.

14. Otherwise, since nothing more can be done with the above procedure we go to the **branch-and-bound** phase. At each node with a new **incumbent solution** we analyze if a substantial number of nonbasic y-variables in the LP solution at the node 1 can be permanently fixed at their current values. The testing is performed by using the reduced costs and the LP optimum at the node 1, and the new incumbent solution as the (stronger) upper bound. If the attempt is successful, we fix the appropriate variables and go to Step 2 to perform again the preprocessing phase but, this time, without probing. At selected branch-and-bound nodes with non-integer solution we proceed as follows: (a) Preprocessing (without probing) to analyzing the feasibility of the partially fixed-and-branched subproblem attached to the node; (b) If the attempt is successful, the node is fathomed; (c) Otherwise, an attempt to obtain a heuristic solution for this subproblem is performed, such that if a new incumbent is found we proceed as before.

15. To prove that the optimal solution of the quasi-global problem is also optimal to the global problem, the incumbent solution is set-up in the quasi-global problem.

16. Next, a testing is performed as in step 7. If it is not successful we go to Step 8; otherwise, the incumbent solution of the quasi-global problem is also optimal for the global problem.

Let us next give some insights on the algorithm outlined above:

1. **Probing procedure**. It is only used during the first execution of the Step 2 (preprocessing phase). It only checks the feasibility of setting $y_{m,f} = 1$ for the two most important processes that are still available per each module; if the result is not successful then $y_{m,f}$ is permanently set to zero. Given the problem's dimensions, a thorough using of probing is expensive but, on the other hand, any type of probing should be performed.

2. The upper bounds B_r (3.3), T_m (3.4) and U_m (3.8) are updated if the following fixing is performed: $y_{m,f} = 1$ and $y_{m1,f} = 0$ for $m1 \in M_f | m1 \neq m$; note also that in the general model (i.e., $\overline{m} > 1$) this fixing implies $pp_{i,n,m} = 1$ and $pp_{i,n,m1} = 0$ for any n such that $p(n) = f$.

3. If the result of the above fixings for $\overline{m} > 1$ is such that a given process, say f can only be performed by at most two modules, say m1 and m2 and $\tilde{l}_{m,f} > T_m$ for both modules, then $y_{m1,f}$ and $y_{m2,f}$ are permanently fixed to one and the bounds B_r , T_m, and U_m are appropriately updated; $\tilde{l}_{m,f}$ is given by (3.12).

4. **Coefficients reduction**. If is always performed for the constraints (3.8), and if $\overline{m} = 1$ then also for the constraints (3.3)-(3.4). In both cases, the constraints are also cleared of common divisors greater than one. Note that each constraint (3.3b) together with its related constraints (3.7b) defines the set of feasible solutions to a knapsack subproblem with special ordered sets; this structure is used for feasibility testing, variables fixing and coefficients reduction. We follow the procedures described in Guignard and Spielberg (1977) and Johnson et al. (1985); they are very effective and practically inexpensive. It is not unusual to find **logical relations** with sum less or equal than a small number; the LP that results may provide a very tight lower bound of the optimal solution.

5. Given the constraints (3.5), the lower bound of the constraints (3.7) must be greater than zero for $\overline{m} > 1$; since it must be integer, it is possible to reduce the range of variability of these constraints. In fact, it will be 2 for the constraint (3.7) related to any process, say f such that any of the conditions (3.16) is satisfied.

$$\nexists m \in M_f \mid \tilde{l}_{m,f} \leq T_m \tag{3.16a}$$

$$\exists r \in R \text{ such that } \nexists m \in M_f \mid \tilde{a}_{r,m,f} \leq B_r \tag{3.16b}$$

where $\tilde{a}_{r,m,f}$ is given by (3.15). Since the value of $\overline{m}$ is not too-high, this reduction is also very interesting.

6. If the heuristic algorithm fails to provide a feasible solution then the quasi-global problem is the same global problem and the upper bound, say $\bar{z}$ of the optimal solution to the global-problem is estimated by increasing in a g% the lower bound, say $\underline{z}$; typically, $g = 25$. Note that $\underline{z}$ is obtained by the LP relaxation of the current quasi-global problem.

7. A great portion of the y-variables may be permanently fixed by the result of **testing the reduced costs** of the LP problem to be solved at the Step 6. The fixing of the variables is performed as follows: fix $y_{m,f} = 0$ and $y_{m,f} = 1$ if the conditions (3.17a) are (3.17b) are satisfied, respectively.

$$\underline{z} + \bar{c}_{m,f} \geq \bar{z} \mid \bar{y}_{m,f} = 0 \tag{3.17a}$$

$$\underline{z} - \bar{c}_{m,f} \geq \bar{z} \mid \bar{y}_{m,f} = 1 \tag{3.17b}$$

where $\bar{y}_{m,f}$ and $\bar{c}_{m,f}$ are the values of the variable and its related reduced cost, respectively.

8. The algorithm goes to the Step 2 from the Steps 10 and 14 if the number of variables, say nf for which the above fixing is performed is such that $nf/nv \geq gv$, where nv gives the number of y-variables still free and gv is a given tolerance; typically, $gv = 0.1$.

9. The **constraints generation** is currently performed by identifying the most violated (lifted) minimal cover of each constraint (3.8). We follow the same procedure for the constraints (3.3)-(3.4) if $\overline{m} = 1$ (see Section 3.3); in this case, we also try to identify new constraints by analyzing k-configurations (see Crowder et al. 1983). See also Van Roy and Wolsey, 1987.

10. We must note that if the solution of two consecutive LP optimizations is such that $(\underline{z2} - \underline{z1})/\underline{z1} < t$ (where $\underline{z2}$ and $\underline{z1}$ give their optimal values and, say $t = 10E\text{-}4$), then the reduced costs fixing and the constraints generation are not performed.

The main strategies for the **branch-and-bound** phase are as follows (we use the well known terminology used by the system MPSX, see the reference):

1. Quasi-integrality tolerance of the y-variables, 0.01.

2. Sub-optimality tolerance of the integer solution, 0.02; i.e., a node is fathomed if its functional value is not better than the 98% of the value of the incumbent solution.

3. The **branching node** is selected according to the criterion of the **best estimation** of the functional value; it is expressed as a linear combination of its functional value and the weighted sum of its integer infeasibilities, such that it can be written

$$F + \alpha \sum_{j} \min \{f(j), 1 - f(j)\} \tag{3.18}$$

where F denotes the functional value at the given node, $f(j)$ gives the fractional part of the current value of the y-variable j at that node, and α is a parameter that can be expressed

$$\alpha = \frac{\bar{z} - \underline{z}}{\sum_{j} \min \{f_1(j), 1 - f_1(j)\}} \tag{3.19}$$

where $f_1(j)$ has the same meaning as above but, now, related to the node 1 (i.e., the LP current quasi-global problem that provides the lower bound $\underline{z}$).

4. The upper bound $\bar{z}$ and the parameter α are recomputed for each new incumbent solution. Recall that $\bar{z}$ at the node 1 is given by the heuristic procedure and, by default, it is estimated as noted above. In any case, **branching priority** is given to the nodes whose functional value is not worse than a given limit such that it can be expressed by $\underline{b} + \beta$, where $\underline{b}$ gives the best functional value of the waiting nodes (note than $\underline{b} = \underline{z}$ at node 1), and β is a dynamically-adjusted parameter. In our case, $\beta = 0.1(\bar{z} - \underline{z})$ but it is gradually increased until every waiting node is a candidate for branching; at the same time that β is increased, higher priority is given to the nodes based on their integer infeasibility by increasing the parameter α. In fact, the value of β is increased by $0.1(\bar{z} - \underline{z})$ and α is doubled if υ consecutive nodes have been scanned without finding a new incumbent solution. The value of υ is a function of the number of y-variables , say ny that take a positive value in the optimal solution of the current LP quasi-global problem, such that $\upsilon = 10 + 0.1ny$.

5. The criterion for selecting the branching variable at each branching node is as follows:

   Case $\overline{m} > 1$. The y-variable to be branched on is selected according to the value of $\tilde{l}_{m,f}$ (3.12), such that greater $\tilde{l}_{m,f}$ higher priority for the related variable.

   Case $\overline{m} = 1$. The **reference row** (see Escudero 1979) for the S1-entity, say f (3.7) is given by the expression

$$\sum_{m \in M1_f} \tilde{l}_{m,f} y_{m,f} \tag{3.20}$$

   where $M1_f$ gives the set of modules that still are available for the process f. The entity S1 to be branched on is selected according to the parameter $b(f)$, where

$$b(f) = \max \{\tilde{c}_{m,f} \ \forall m \in M1_f\} \text{ for (3.2)} \tag{3.21a}$$

$$b(f) = \max \{\tilde{l}_{m,f} \ \forall m \in M1_f\} \text{ for (3.13)} \tag{3.21b}$$

   such that greater $b(f)$ higher priority for the S1-entity f.

Obtaining the initial solution (For more details, see Escudero 1986).

Case $\overline{m} > 1$. The problem (3.2)-(10) (and the same with (3.13)) is decomposed in three sub-problems. First, we obtain the set of modules that are to be assigned to each process, by using an ad-hoc heuristic (greedy) procedure such that there is at least one feasible solution; let $M1_f$ (res. $F1_m$) denote the set of modules (res. process) that are allocated for each process (res. module). Second, we improve the above solution by solving the **Linear Transportation Problem** (hereafter, LTP) with **positive gains**: min (3.2) subject to (3.4)-(3.5) and (3.10), where the sets M_f and F_m are substituted by the sets $M1_f$ and $F1_m$, respectively. Third, an ad-hoc re-arrangement of $\{pp_{i,n,m}\}$ is performed such that the violated constraints (3.3) and (3.6) are satisfied.

Case $\overline{m} = 1$. First, solving the **Generalized Assignment Problem**: min (3.2) subject to (3.7)-(3.9). Second, if the constraints (3.3)-(3.4) are satisfied, it is the optimal solution; otherwise, an appropiate re-arrangement of $\{y_{m,f}\}$ is performed. Note that the algorithm may exploit the structure of the constraints (3.8), where the coefficient u_f does not depend on m; see Sandi (1975).

3.5. COMPUTATIONAL EXPERIENCE

The algorithm was tested by implementing a prototype running on the IBM 4381-P13 with VM/SP-4.0; it was written in PL/I and compiled with the version 1.3. It uses interactively the system MPSX-MIP/370 via ECL. Table 1 reports some computational experience with six real-life cases. P1 to P6 require $\overline{m} = 4$. P7 and P8 are as P1 and P2, respectively but $\overline{m} = 1$. The objective function to minimize is the resource cost (3.2a).

All runs have a $q = 2\%$ sub-optimality tolerance; i.e., a node will be fathomed if its functional value, say F is such that $F \geq 0.98\overline{z}$, where $\overline{z}$ is the value of the incumbent solution. GAP denotes $(\overline{z} - \underline{z})/\underline{z}$, where $\underline{z}$ now gives the lowest functional value (note that it is not necessarily the value of the LP relaxation). Then, $GAP \geq 2\%$ and, usually, it is smaller. The table reports the number of constraints and variables for the global problem; it gives an indication of the case's size. However, the size of any quasi-global problem (whenever it can be used) is smaller; note that $y_{m,f} = 0$ implies $pp_{i,n,m} = 0$ for $\forall n \in N_i | p(n) = f, i \in I$. The heuristic algorithm finds in all cases a feasible solution and, then, only quasi-global problems were optimized for P1 to P6; each quasi-global problem was revised once, but the number of fixed variables never was less than 33.4%. The reported time is the total CPU time including preprocessing, but excluding I/O operations.

Additionally, we have solved till optimality the case P4. The optimal value was very similar to the value obtained for the 2% sub-optimal solution, but it requires 7.58 minutes; i.e., almost 100% of additional computational effort was needed for improving the solution and proving optimality. It is in accordance with other experiences for smaller problems. Usually, the gain if any is not balanced with the computational time that it requires. We have also experimented running the case till optimality, but without using the solution provided by the heuristic algorithm, nor any type of preprocessing; then, the full global problem is to be used; it required 12.47 minutes for obtaining the optimal solution, and its optimality was not yet proved after 20 minutes of CPU time.

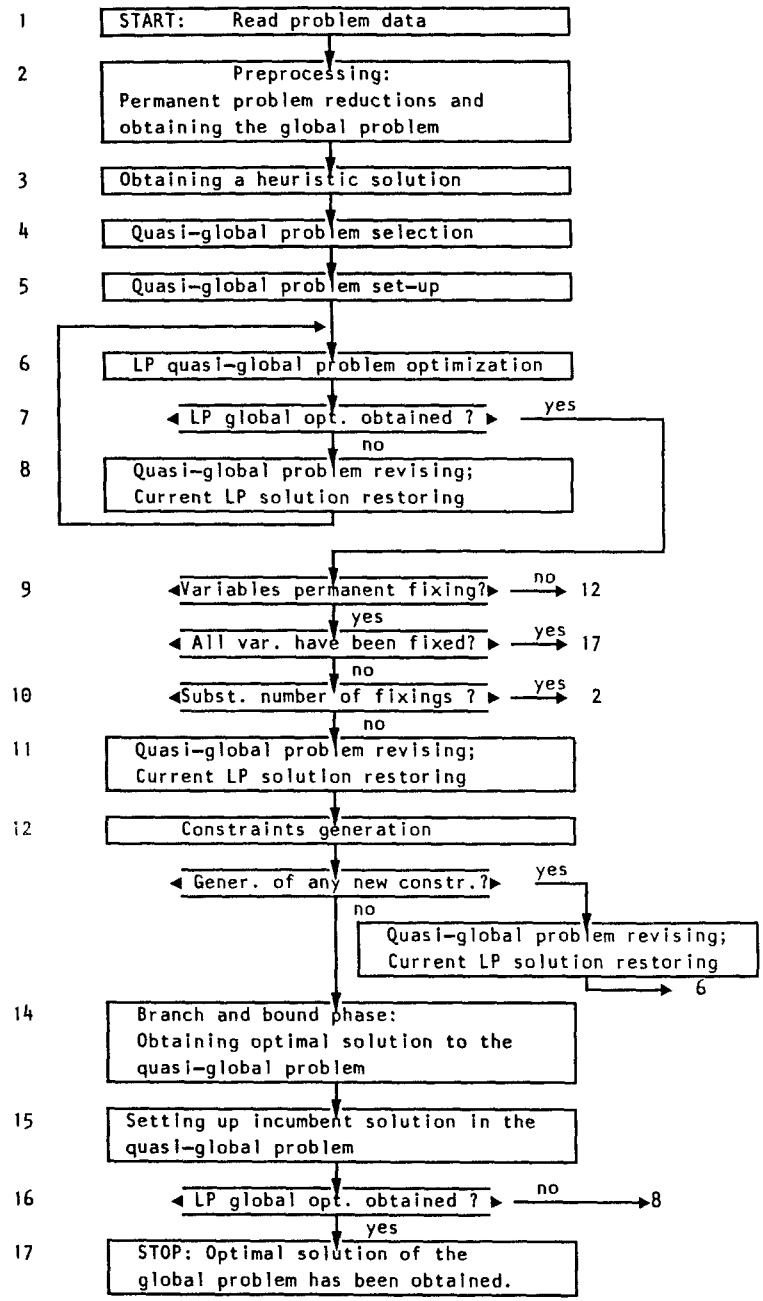


Figure 3. The overall flowchart

Table 1. Computational results. Modules Workload Assignment.

Parameters	P1	P2	P3	P4	P5	P6	P7	P8
Number part types	5	8	8	8	9	10	5	8
Average number operations per part type	15	25	28	39	43	31	15	25
Number modules	5	6	10	10	10	15	5	6
Number processes	12	21	25	33	36	27	12	21
Number resources	9	13	9	10	10	10	9	13
Constraints	218	473	666	972	1006	736	23	34
Cont. variables	392	1201	2157	3121	3671	2748	-	-
0-1 variables	45	117	210	286	312	267	45	113
Density (%)	1.9	0.9	0.7	0.5	0.4	0.7	10.8	5.62
Branch-and-bound:								
-Solutions	3	2	3	2	3	1	3	2
-Total number nodes	13	58	68	228	180	16	31	16
GAP (%)	0.0	0.32	0.00	2.00	0.0	1.62	0.00	0.41
Time (m)	0.13	0.59	1.56	3.87	4.75	2.38	0.10	0.05

4. OPERATIONS EXECUTION SEQUENCING AND PART TYPE ROUTING

4.1. PLANNING TASK SUPPORTED BY THE MODEL

Obtaining the sequence $\{so_h|i\}$ for the operations execution, and the routing proportion $\{rp_{m1,m2}|n1 \rightarrow n2|i\}$ that must be sent through each inter-modules path for each part type in the set I. Let **Feasible Sequencing Ordering** (hereafter, FSO) denote any complete operations ordering $\{n1 \rightarrow n2\}$ that is consistent with the direct and immediate precedence relationships given by the set $A_{i,1} \cup A_{i,2}$, and does not violate the restrictions on the feasibility of the inter-modules routing. The goal consists of obtaining the FSO for each part type such that the transport cost/time is minimized. Since the transport matrix Q is not related to the part types, and the graphs of precedence relationships $\{G_i\}$ are independent, let us drop the subindex i from the notation given in Section 2.

4.2. ROUTING PROPORTIONS FOR THE POTENTIAL PARTIAL ORDERING $n1 \rightarrow n2$

Let P denote the **reachable** matrix in the graph G, such that its element $p_{n1,n}$ will have the value 1 if n is reachable from $n1$, i.e., there is, at least, one predecessor path from the node n to the node $n1$; otherwise, $p_{n1,n} = 0$. Let the matrix S be such that $s_{n1,n2} = 1$ if it has not yet been detected the infeasibility of the potential partial ordering $n1 \rightarrow n2$ in any FSO and, otherwise, it is zero for $n1,n2 \in N$. Then, initially, $s_{n1,n2} = 1$ if any of the conditions (4.1) is satisfied.

$$(n1,n2) \in A_1 \cup A_2 \qquad (4.1a)$$

$$p_{n1,n2} = 0 \wedge p_{n2,n1} = 0 \wedge \nexists n \in N | (n,n2) \in A_2 \wedge \nexists n \in N | (n1,n) \in A_2 \quad (4.1b)$$

The routing proportions $\{rp_{m1,m2} | n1 \rightarrow n2\}$ must be such that the transport cost/time, say $\{c_{n1,n2}\}$ is minimized. It is obtained by solving the LTP (4.2)-(4.5) for each potential partial ordering $n1 \rightarrow n2$. (Note: $c_{n1,n2} = \infty | s_{n1,n2} = 0$ and, then, the LTP is not required). Let us drop $n1 \rightarrow n2$ from the rp-variables and let, now, $M_{p(n)} = \{m \in M | pp_{n,m} > 0\}$ for $n \in \{n1,n2\}$. (The production proportions $\{pp_{n,m}\}$ are given by the model **Modules workload assignment**).

$$\min \; c_{n1,n2} = \sum_{m1 \in M_{p(n1)}} \sum_{m2 \in M_{p(n2)}} q_{m1,m2} rp_{m1,m2} \quad (4.2)$$

subject to

$$\sum_{m2 \in M_{p(n2)} | q_{m1,m2} \neq \infty} rp_{m1,m2} = pp_{m1,m2} \quad \forall m1 \in M_{p(n1)} \quad (4.3)$$

$$\sum_{m1 \in M_{p(n1)} | q_{m1,m2} \neq \infty} rp_{m1,m2} = pp_{m1,m2} \quad \forall m2 \in M_{p(n2)} \quad (4.4)$$

$$rp_{m1,m2} \geq 0 \quad \forall m1,m2 \quad (4.5)$$

Note that the model can be solved by inspection if $|M_{p(n1)}| = 1 \vee |M_{p(n2)}| = 1$ as it frequently happens. In any case, it has small dimensions. The matrix S is updated by (4.6) if the LTP is infeasible.

$$s_{n1,n2} := 0 \mid s_{n1,n2} = 1 \wedge c_{n1,n2} = \infty \quad (4.6)$$

4.3. MODELIZATION

Let $N2(n1)$ (res. $N1(n2)$) define the set of nodes for which it has not yet been detected that they cannot be immediate successors (res. predecessors) of the node $n1$ (res. $n2$) in any FSO.

$$N2(n1) = \{n2 \in N | s_{n1,n2} = 1\} \quad \forall n1 \in N \quad (4.7a)$$

$$N1(n2) = \{n1 \in N | s_{n1,n2} = 1\} \quad \forall n2 \in N \quad (4.7b)$$

The so-called **Sequential Ordering Problem** (hereafter, SOP) consists of obtaining the FSO that minimizes the penalty given by the matrix $C = \{c_{n1,n2}\}$. The penalty, say z(SO) associated with a given **Sequential Ordering** (hereafter, SO) can be expressed as follows

$$z(SO) = \sum_{n1 \in N} \sum_{n2 \in N2(n1)} c_{n1,n2} y_{n1,n2} \quad (4.8)$$

where $y_{n1,n2}$ is a 0-1 variable such that $y_{n1,n2} = 1$ if $n1 \rightarrow n2$ and, otherwise, it is zero. Then, the problem consists of

$$\min \; \{z(SO) \mid \text{SO is a FSO}\} \quad (4.9)$$

There is not a unique formulation of SOP (4.8)-(4.9). The formulation that seems to be most attractive can be more easily expressed if we require without loss of generality that there is only one node, say 0 such that $\nexists n \in N | (n,0) \in A_1 \cup A_2$; then, the SOP can be expressed as the ATSP with side (precedence relations based) constraints.

Let $h1_n$ and $h2_n$ $\forall n \in N$ denote the lowest and highest sequencing levels of the node n in any FSO, respectively; then, $h1_n = 1 + |P1_n|$ and $h2_n = |N| - |P2_n|$, where

$$P1_n = \{n1 \in N | p_{n1,n} = 1\} \quad (4.10a)$$

$$P2_n = \{n2 \in N | p_{n,n2} = 1\} \quad (4.10b)$$

Let $x_{n,h}$ be a 0-1 variable such that $x_{n,h} = 1$ if the node n is sequenced at the level h and, otherwise, it is zero; let $y_{n1,n2}$ be also a binary variable with the same meaning as in (4.8). Then,

$$x_{n,h} \in \{0;1\} \text{ for } h = h1_n,\ldots, h2_n \text{ and, otherwise, } x_{n,h} = 0 \tag{4.11}$$

$$y_{n1,n2} \in \{0;1\} \ \forall n1,n2 \in N | s_{n1,n2} = 1 \tag{4.12}$$

Note that $so_h \neq n | x_{n,h} = 0$ in any FSO, and $s_{n1,n2} = 1$ implies that there is at least one level, say h such that $h1_{n1} \leq h \leq h2_{n1}$ and $h1_{n2} \leq h + 1 \leq h2_{n2}$. The SOP can be formulated as follows

$$\min z = \sum_{n1,n2 \in N} c_{n1,n2} y_{n1,n2} \tag{4.13}$$

subject to (4.11)-(4.12), and

$$\sum_{n2 \in N2(n1)} y_{n1,n2} = 1 \quad \forall n1 \in N \tag{4.14}$$

$$\sum_{n1 \in N1(n2)} y_{n1,n2} = 1 \quad \forall n2 \in N \tag{4.15}$$

$$\sum_{n1,n2 \in T} y_{n1,n2} \leq |T| - 1 \quad T \subset N, \quad 2 < |T| < |N| \tag{4.16}$$

$$\sum_{h \in N} x_{n,h} = 1 \quad \forall n \in N \tag{4.17}$$

$$\sum_{n \in N} x_{n,h} = 1 \quad \forall h \in N \tag{4.18}$$

$$\sum_{t1 \in N} t1 \times x_{n1,t1} + 1 \leq \sum_{t2 \in N} t2 \times x_{n2,t2} \quad \forall (n1,n2) \in A_1 \tag{4.19}$$

$$x_{n1,h} + x_{n2,h+1} \leq 1 + y_{n1,n2} \quad \forall n1,n2 \in N | s_{n1,n2} = 1, \ h = 1,2,\ldots,|N| - 1 \tag{4.20}$$

$$x_{n1,h} + x_{n2,h+1} \leq 1 \quad \forall n1,n2 \in N | s_{n1,n2} = 0, \ h = 1,2,\ldots,|N| - 1 \tag{4.21}$$

$$x_{n1,h} = x_{n2,h+1} \quad \forall (n1,n2) \in A_2, \ h = 1,2,\ldots,|N| - 1 \tag{4.22}$$

Two blocks can be recognized in the above model. The constraints (4.14)-(4.16) are related to the ATSP, and (4.17)-(4.22) are related to the precedence relationships. (4.14) (res. (4.15)) avoid more than one partial ordering $n1 \rightarrow n2$ for any $n1$ (res. $n2$), and (4.16) avoid subtours. (4.17) (res. (4.18)) avoid more than one level (res. operation) for a given operation (res. level). (4.19) prevent that an operation have a lower sequencing level than the sequencing level of any operation that must be directly precedent. (4.20) force the penalty $c_{n1,n2}$ if $n1 \rightarrow n2$ belongs to the solution. (4.21) avoid the partial ordering $n1 \rightarrow n2$ if it is not allowed in any FSO. (4.22) force $n1 \rightarrow n2$ for any immediate precedence relationship.

4.3. ALGORITHMIC APPROACH

Solving the ATSP with side constraints given in Section 4.2 may require more CPU time than allowed. If the optimality is required the best approach seems to be using a cutting planes based LP relaxation for obtaining a tight lower bound of the optimal solution and fixing variables and, next, using the branch-and-bound methodology for obtaining the optimal solution. See in Section 3.3 a detailed framework for the LP-based combinatorial problem solving.

The proposed (inexact) algorithm for solving SOP (4.8)-(4.9) is as follows (For more details, see Escudero 1987):

1. Obtaining the initial sets $N2(n)$ and $N1(n)$ $\forall n \in N$ by using the information provided by the set $A_1 \bigcup A_2$ and the matrix C.

2. **SOP's preprocessing**. A strong characterization of the potential partial orderings $\{n1 \rightarrow n2\}$ is performed, such that the feasibility of any $n1 \rightarrow n2$ is analyzed; i.e., the goal consists of reducing as much as possible the cardinality of the sets $N2(n)$ and $N1(n)$.

3. **Obtaining a lower bound**, say $\underline{z}$ of the optimal solution to SOP. It is obtained by solving the **Assignment Problem** (hereafter, AP) that results from the relaxation of the constraints (4.16) in the ATSP (4.12)-(4.16). Let $\{\bar{y}_{n1,n2}\}$ denote the value of the variables in the optimal solution. We use the algorithm so-called CTCS given by Carpaneto et al. (1985).

4. **Reducing the number of existing subtours**, if any. We use a modification of the **Patching Algorithm** (hereafter, PA) given by Karp and Steele (1985). In this step, we only look for those 'patchs' that do not violate the precedence relationships and do not deteriorate the value of the objective function (i.e., only alternative solutions to the optimal one to AP are considered). If there are no more subtours, the solution is optimal.

5. **Tightening the lower bound** $\underline{z}$. We use the bounding procedures BP1, BP2 and BP3 as given by Balas and Christofides (1981). The related **admissible graph**, say $G_0 = (N, A_0)$ is also obtained. The graph G_0 is a subgraph of G containing those and only those arcs with zero reduced cost, i.e.,

$$A_0 = \{(n1,n2) \text{ such that } n1,n2 \in N | \bar{c}_{n1,n2} = 0\} \tag{4.23}$$

where $\bar{c}_{n1,n2}$ is the reduced cost of the arc $(n1,n2)$ such that

$$\bar{c}_{n1,n2} = c_{n1,n2} - u_{n1} - v_{n2} - \sum_{t \in T} w_t a_{t,n1,n2} \tag{4.24}$$

u_{n1} and v_{n2} for $n1,n2 \in N$ are the dual variables associated with the constraints (4.14) and (4.15), respectively; $a_{t,n1,n2}$ is the coefficient of $y_{n1,n2}$ in the ATSP constraint, say t that is violated by the optimal solution to AP, w_t gives its related dual variable, and $T = T1 \bigcup T2 \bigcup T3$, where T1, T2 and T3 are the three types of inequalities that are used by the bounding procedures BP1, BP2 and BP3, respectively. The sets T1, T2 and T3 are equivalent to the constraints (4.16).

6. **Obtaining a Hamiltonian circuit** (hereafter, $\mathcal{H}$) . Let the so-called **augmented admissible graph**, say $\tilde{G}_0 = (N, \tilde{A}_0)$ be such that $\tilde{A}_0 = A_0 \bigcup \{(n1,n2)\}$, where the set $\{(n1,n2)\}$ gives the arcs with the smallest positive reduced cost in the admissible graph G_0; i.e., the arc $(n1,n2)$ must be such that $\bar{c}_{n1,n2} = \underline{c}$, where

$$\underline{c} = min\{\bar{c}_{n1,n2} \ \forall (n1,n2) \notin A_0 | s_{n1,n2} = 1\} \tag{4.25}$$

We use the implicit enumeration algorithm given by Martello (1983) for obtaining $\mathcal{H}$; the parameters are as follows: $nc = -1$ (i.e., only one $\mathcal{H}$ is requested) and $nb = 20$ (maximum number of allowed backtrackings). Note that if all arcs in $\mathcal{H}$ have null reduced cost and do not violate the precedence relationships given by the set $A_1 \bigcup A_2$, then $\mathcal{H}$ is an optimal solution to SOP. Let $\{\ddot{y}_{n1,n2}\}$ denote the value of the variables in $\mathcal{H}$.

7. **Testing the feasibility of** $\mathcal{H}$, if any. By definition, $\mathcal{H}$ is a feasible solution to ATSP, but it could be a non-feasible one to SOP. In this case (i.e., $\exists n1,n \in N | p_{n1,n} = 1$ such that there is a successor path from n to $n1$ in $\mathcal{H}$ that does not pass through the node 0), the nodes n and $n1$ are permuted in $\mathcal{H}$. Note that if there is, at least, one such a permutation then a new feasibility testing must be performed by using the matrices P and S. Let $\bar{z}$ denote the value of the objective function (4.8) for the feasible $\mathcal{H}$. It is assumed that a q-sub optimal solution to SOP has been obtained if $(\bar{z} - \underline{z})/\underline{z} \leq q\%$; currently, $q = 3$.

8. **Reduced cost fixing**. By using the reduced cost (4.24) of the potential partial orderings, the matrix S can be updated, such that

$$s_{n1,n2} := 0 | \ s_{n1,n2} = 1 \ \wedge \ \bar{c}_{n1,n2} \geq \bar{z} - \underline{z} \ \wedge \ \ddot{y}_{n1,n2} = 0 \tag{4.26}$$

Note that $\bar{c}_{n1,n2} = 0$ for $\bar{y}_{n1,n2} = 1$. If, as the result of (4.26), $|N2(n1)| = 1 \vee |N1(n2)| = 1$ then perform a new preprocessing as in Step 2. If, as a result of any further fixing, the condition (4.27) is satisfied then go to the Step 3; a tighter lower bound could be found.

$$\exists n1,n2 \in N | \bar{y}_{n1,n2} = 1 \wedge s_{n1,n2} = 0 \tag{4.27}$$

9. **Eliminating the existing subtours**, if any. We use a modification of PA, such that the feasible solution to ATSP is also a feasible one to SOP. In this step, we use the costs $\{c_{n1,n2}\}$. It is expected that the value of the objective function is not strongly deteriorated while keeping feasible the solution. If a FSO has not yet been obtained, an interactive graphic approach is used for eliminating the remaining subtours; alternatively, an implicit enumeration methodology can be used (see Chatterjee et al., 1984). A new reducing cost fixing is performed as in (4.26) if the new solution, if any is better than the previous one.

10. **Improving the FSO**. We use a modification of the local search described by Kanellakis and Papadimitrou (1980) for the ATSP, such that the new solution is a feasible one to SOP.

11. **Reduced cost fixing** if the FSO has been improved. The result is given to the planner as a (hopefully) useful information for his potential attempt to improve the FSO offered by the algorithm.

For illustrative purposes, let the case shown in Figure 1a and Tables 2 and 3 with $|N| = 7$ operations and $|M| = 4$ modules. The matrices P, S and C are shown in Table 4. The results are as follows: $\underline{z} = 13.50$ from AP; $\underline{z} = 14.10$ from the bounding procedure, and $\mathcal{H}$: (0,1,4,3,6,2,5,7,0) with $\bar{z} = 14.10$ and it is a FSO; then it is the optimal solution. The optimal routing proportions are shown in Table 5 and Figure 4.

4.4. COMPUTATIONAL EXPERIENCE

The algorithm was tested by implementing a prototype running on the IBM 4381-P13 with VM/SP-4.0; it was written in PL/I and compiled with the version 1.3, except the codes AP and $\mathcal{H}$ that were written in Fortran (see Carpaneto et al 1985, and Martello 1983) and compiled with the version VS 1.4. Table 6 reports some computational experience. Up to 6 modules are considered and all inter-module transport segments are allowed. The problems belong to nine classes based on sizes with $|N| = 8$, 31, 36, 58, 68, 71, 84, 91 and 98 operations; each class is included by one real-life and ten generated problems. The latter ones just only obtain the FSO; instead of calculating the penalty matrix C by solving a LTP for obtaining the optimal routing proportions of each potential partial ordering, the coefficients $c_{n1,n2}$ were obtained as follows: $c_{n1,n2} = \infty$ if so in the related real-life problem and, otherwise, it was drawn from a uniform distribution with the range $[0.9 \times c_{n1,n2},\ 1.1 \times c_{n1,n2}]$.

The headings of the table are as follows: *NPR*, number of (direct) precedence relationships. *nLTP* , number of LTP's solved for the real-life problem. *T up LTP* , CPU time (secs.) (I/O operations are not counted) for obtaining the matrices P, S and C, including preprocessing. *T* , time (secs.) required for solving the SOP; *aver* , average time of the eleven entries. *OPT* , number of cases where the algorithm detects the optimality of the FSO. $GAP = (\bar{z} - \underline{z})/\underline{z}$; *aver*: the two worst cases per each class are not included, nor the cases whose solution was proved to be optimal. *NR* , total number of potential partial orderings (i.e., number of elements with value 1 in the matrix S before performing the SOP's preprocessing). *R* , number of potential partial orderings that by average were removed by the SOP's preprocessing and the reduced costs fixing. Note: the counting of R does not include the cases with optimal FSO.

Based on the results shown in Table 6, it seems that the performance of the algorithm is very similar for the real-life and generated problems of each class. We can see that the CPU time increases almost linearly on the number of nodes; the range of the CPU time is very small within each class. Note also that the maximum time required for solving any problem was 31.25 seconds, and the algorithm found a FSO in all problems. Only in one case the planner could improve the solution offered by the algorithm; the improvement amounted to 1.8% on the value of the objective function. It seems that it could be extremely difficult to do so, should he not be provided with the graphical representation of the graph of precedence relationships,

the augmented admissible graph, and the graph showing the potential partial orderings $\{n1 \rightarrow n2\}$ that cannot be included in any better solution than the solution offered by the algorithm. As it was expected, the solution's optimality was proved only for few cases; however, we have detected by inspection cases where the FSO offered by the algorithm was optimal. In any case, the gap between the FSO and the lower bound of the optimal solution is not too-big, and the percentage of potential partial orderings removed on the average is not too-small.

Table 2. Matrix Q

\m2 \ m1\	1	2	3	4
1	0	1	2	3
2	4	0	5	6
3	7	8	0	9
4	10	11	12	0

Table 3. Production proportions {ppn,m} %

\ m \ n \	1	2	3	4
1	25	25	25	25
2	50	25	50	0
3	0	60	0	40
4	10	20	20	50
5	40	30	30	0
6	0	100	0	0
7	0	0	100	0

Table 4. Matrices P, S and C.
Case given by Figure 1a and Tables 2 and 3

\n2 \ n1\	1	2	3	4	5	6	7
1	0;0;-	1;1;250	1;1;240	1;1;120	1;0;-	0;1;500	0;1;475
2	0;0;-	0;0;-	0;1;305	0;1;195	1;1;15	0;1;250	0;1;225
3	0;0;-	0;1;565	0;0;-	0;1;200	1;1;550	0;1;440	0;1;780
4	0;0;-	0;1;515	0;1;280	0;0;-	1;1;530	0;1;720	0;1;720
5	0;0;-	0;0;-	0;0;-	0;0;-	0;0;-	0;1;280	0;1;230
6	0;1;375	0;1;325	0;1;240	0;1;440	0;1;310	0;0;-	0;1;500
7	0;1;600	0;1;550	0;1;840	0;1;680	0;1;520	0;1;800	0;0;-

Each element must be read: $p_{n1,n2}$; $s_{n1,n2}$; $c_{n1,n2}$ (in 100.)

Table 5. Optimal routing proportions $\{r_{pm1,m2|n1 \rightarrow n2}\}$ (%) for the case given in Table 4

1 → 4

\m2 \ m1\	1	2	3	4
1	10			15
2		20		5
3			20	5
4				25

4 → 3

\m2 \ m1\	2	4
1	10	
2	20	
3	20	
4	10	40

3 → 6

\m2 \ m1\	2
2	60
4	40

6 → 2

\m2 \ m1\	1	2	3
2	50	25	25

2 → 5

\m2 \ m1\	1	2	3
1	40	5	5
2		25	
3			25

5 → 7

\m2 \ m1\	3
1	40
2	30
3	30

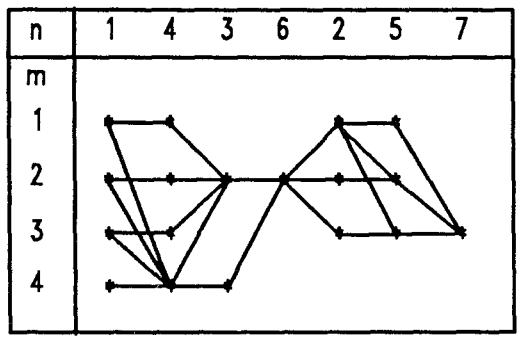


Figure 4. Optimal routing. See Table 5

Final remarks

The proposed algorithm does not attempt to guarantee the solution's optimality, although sometimes it does so. High cost range and tight precedence relationships often produce high GAP. It seems that bounding procedures that use the information provided by the matrix P should be investigated; we are planning as future research to analyze the performance of

using cutting planes LP based lower bounds (a good balance between the tightness of the lower bound and the computing effort is needed); identifying violated precedence relationships is not a big problem, but deciding what variables are to be left free in the successive LP optimizations and what constraints are to be included as cutting planes is up to now an open problem. Other topic for future research consists of analyzing the performance of ATSP as a provider of lower bounds for SOP. In any case, based on our computational experience and the by-hand results obtained by the planner, it seems that the algorithm can be a useful tool for producing good FSO's almost on-line, as it is frequently required when planning the FMS's utilization.

Table 6. Computational results for the SOP (11 entries per class)

class	|M|	|N|	NPR	nLTP	T up LTP	T		OPT	GAP (%)		T/|N| x 100	R/NR (%)
						aver	max		aver	max		
1	4	8	8	34	0.40	0.39	0.42	3	2.3	8.9	5	98.4
2	3	31	172	66	0.63	1.62	1.82	2	3.4	22.8	5	89.3
3	6	36	187	120	1.02	2.12	2.54	0	5.6	27.5	6	85.1
4	4	58	142	227	1.51	5.86	6.17	0	4.1	15.4	10	78.2
5	3	68	236	346	2.24	7.63	8.09	1	3.0	8.2	11	74.3
6	4	71	327	510	3.65	7.47	7.92	0	4.3	11.6	10	86.5
7	5	87	560	419	4.24	2.59	13.29	0	6.8	10.7	14	84.6
8	5	90	468	631	7.43	3.21	15.78	1	5.6	28.8	14	88.7
9	5	98	521	945	9.70	8.38	21.55	0	9.7	37.8	18	66.1

5. PARTS LOADING SEQUENCING AND PROCESSING ROUTE

5.1. PLANNING TASK SUPPORTED BY THE MODEL

Obtaining the loading sequence $\{lo_j\}$ of the parts to be processed in the FMS along the given planning horizon, and the processing route $\{pr_{j,n}\}$ for each part. See related approaches in Wittrock (1985 and 1986), and Greene and Sadowski (1986).

In addition to the definitions given in Section 2, let $\bar{tl}_m$ (5.1) and $\hat{tl}$ (5.2) denote the **total workload of the module** m for $m \in M$, and the **total workload of the FMS**, respectively. These elements are provided by the model **Modules workload assignment** (see Section 3).

$$\bar{tl}_m = \sum_{l \in I} \sum_{n \in N_l} \tilde{t}_{i,n,m} D_l pp_{i,n,m} \tag{5.1}$$

$$\hat{tl} = \sum_{m \in M} \bar{tl}_m \tag{5.2}$$

where

$$\tilde{t}_{i,n,m} = \frac{t_{i,n}}{e_{m,p(n)}} \tag{5.3}$$

Let the following elements for the j^{th} loading level for $j = 1,2,...,|J|$: $al_{m,j}$ (5.4), the **actual workload of the module** m; tl_j (5.5), the **actual workload of the whole FMS**; and $el_{m,j}$ (5.6), the **expected workload of the module** m,

$$al_{m,j} = \sum_{g \in J_j} \sum_{n \in N_{lo(g)} | p(n) \in F_m} \bar{t}_{lo(g),n,m} y_{g,m,n} \tag{5.4}$$

where J_j denotes the set of loading levels $\{1,2,...j\}$, and $y_{g,n,m}$ is a 0-1 variable such that its value will be one if the operation n to be executed in the part loaded in the g^{th} level is performed by the module m; otherwise, it is zero. I.e., $y_{g,n,m} = 1 \equiv pr_{g,n} = m$.

$$tl_j = \sum_{m \in M} al_{m,j} \tag{5.5}$$

$$el_{m,j} = \frac{\bar{tl}_m}{\hat{tl}} tl_j \quad \forall m \in M, j \in J \tag{5.6}$$

Note that the set $\{el_{m,j}\}$ gives the 'ideal' workload for balancing the utilization of the modules along the time units of the planning horizon.

The parts loading sequencing and the processing route for each part must be such that the **total absolute deviation** (5.7) of the actual workload $\{al_{m,j}\}$ from the expected workload $\{el_{m,j}\}$ along the time units of the planning horizon is minimized, and each part type is processed according to the given routing proportions and operations execution sequence. (These elements are provided by the model **Operations execution sequencing and part type routing**).

$$\min \{\sum_{j \in J} d_j\} \tag{5.7}$$

where

$$d_j = \sum_{m \in M} |al_{m,j} - el_{m,j}| \tag{5.8}$$

(Similar approach could be used for minimizing the **greatest absolute deviation**). The goal consists of reducing the turnaround time per each part and, then, reducing the WIP inventory.

5.2 MODELIZATION

The dimensions of the y-variables in (5.4) are defined by $\forall m \in M_{p(n)}$, $n \in N_{lo(g)}$, and $g \in J$. Besides, (5.7) may require additional instrumental variables. Given the problem's dimensions, a classical mathematical programming formulation is not practical for our purposes and, then, is omitted here. As an academic exercise, see in Escudero (1986 pp.69-72) the modelization of the constraints of a feasible loading sequence.

Although we cannot effort to use exact algorithms, a 'local' minimization of the function (5.7) still may produce good results and the computational effort is affordable. Let z_i denote the **smallest total absolute deviation** of the actual workload from the expected workload at the level j, provided that $lo(j) = i$; note that the unknown is the **processing route** $\{pr_{j,n}\}$ of the part to be loaded. Then, instead of minimizing (5.7), we only try to obtain the type $lo(j)$ at the level j for $\forall j \in J$ such that

$$lo(j) \;=\; arg.min\ \{ z_i \ \forall i \in \bar{I}_j \} \tag{5.9}$$

where $\bar{I}_j$ gives the set of part types whose production volume has not yet been completely loaded. Note, then, that we obtain **at each loading level** the best part to be loaded without modifying the previous loading levels. Let $\{y_{n,m}\}$ and $\{x_{h,m1,m2}\}$ be 0-1 variables; $y_{n,m}$ will be one if the operation n is performed by the module m and, otherwise, it is zero for $\forall m \in M_{p(n)}$, $n \in N_i$; $x_{h,m1,m2}$ will be one if the operations $n1$ and $n2$ are executed by the modules $m1$ and $m2$, respectively and, otherwise, it is zero for $\forall m1 \in M_{p(n1)}$, $m2 \in M_{p(n2)}$ and $n1 \rightarrow n2$,

being $n1 = so_h|i$ for $h = 1,...,|N_i| - 1$, such that $rp_{m1,m2} > 0|n1 \rightarrow n2|i$. Then, z_i is the solution of the problem (5.10)-(5.17).

$$z_i = \min \{ \sum_{m \in M} |al_{m,j} - el_{m,j}| \} \tag{5.10}$$

where

$$al_{m,j} = \sum_{g \in J_{j-1}} \sum_{n \in N_{lo(g)} | pr_{g,n} = m} \bar{t}_{lo(g),n,m} + \sum_{n \in N_i | p(n) \in F_m} \bar{t}_{i,n,m} y_{n,m} \tag{5.11}$$

where $\bar{t}_{i,n,m}$ is given by (5.3). The expression for $el_{m,j}$ is given by (5.6); it uses (5.1), (5.2), (5.5) and (5.11). The minimization (5.10) is subject to

$$\sum_{m \in M_{p(n)}} y_{n,m} = 1 \quad \forall n \in N_i \tag{5.12}$$

$$\sum_{m2 \in M_{p(n2)} | rp_{m1,m2} > 0 | n1 \rightarrow n2|i} x_{h,m1,m2} = y_{n1,m1} \quad \forall m1 \in M_{p(n1)},\ n1 = so_h|i \tag{5.13}$$

$$\sum_{m1 \in M_{p(n1)} | rp_{m1,m2} > 0 | n1 \rightarrow n2|i} x_{h,m1,m2} = y_{n2,m2} \quad \forall m2 \in M_{p(n2)},\ n2 = so_{h+1}|i \tag{5.14}$$

$$x_{h,m1,m2} \in \{0;1\},\ \ y_{n,m} \in \{0;1\} \tag{5.15}$$

$$x_{h,m1,m2} = 0 \ | \ \eta_{m1,m2}|n1 \rightarrow n2|i|j = 0 \tag{5.16}$$

where $h = 1,...,|N_i| - 1$. And, $\eta_{m1,m2}|n1 \rightarrow n2|i|j$ for $n1 = so_h|i$ gives the number of parts of type i that, at the loading level j, still may be sent from the module $m1$ to the module $m2$ for executing the operations $n1$ and $n2$, respectively; it is given by the expression

$$(rp_{m1,m2}|n1 \rightarrow n2|i)D_j - |L| \tag{5.17a}$$

where

$$L = \{g \in J_{j-1} | lo(g) = i \wedge m1 = pr_{g,n1} \wedge m2 = pr_{g,n2}\} \tag{5.17b}$$

Constraints (5.12) force that each operation be executed by just only one module. For the module that has been selected, say $m1$ (res. $m2$) for the operation $n1$ (res. $n2$), (5.13) (res. (5.14)) force an appropriate module, say $m2$ (res. $m1$) where the operation $n2$ (res. $n1$) is to be executed for any $n1 \rightarrow n2$. (5.16) prevent the using of any path whose 'capacity' has already been used by the previous levels. Note that $x_{h,m1,m2} = 1$ implies that $y_{n1,m1} = 1$ and $y_{n2,m2} = 1$ for $n1 = so_h|i$ and $n1 \rightarrow n2$ and, then, $pr_{j,n1} = m1$ and $pr_{j,n2} = m2$, provided that $lo(j) = i$. Note: some provisions must be made in the above formulation for guaranteeing that, computationally, (5.17) gives an integer number and the condition (5.18) is satisfied (see Figure 4).

$$\sum_{m1 \in M_{p(n1)}} (rp_{m1,m}|n1 \rightarrow n|i)D_j = \sum_{m2 \in M_{p(n2)}} (rp_{m,m2}|n \rightarrow n2|i)D_j \quad \forall m,n,i \tag{5.18}$$

5.3. ALGORITHMIC APPROACH

Note that even the optimal solution to (5.10)-(5.17) may require in some cases more computing time than it could be afforded. See in Escudero (1986 pp.73-75, 121-127) the details, but the main steps of the (inexact) algorithm that currently we are using are as follows:

1. **Obtaining a feasible processing route**, say $\{pr_{j,n}\}$ for the loading level j, provided that $lo(j) = i$, such that $\bar{z}_i$ will denote the distance to be used in (5.9) instead of the optimal one z_i (5.10). Let us consider the direct graph $\bar{G}$ (see Figure 4), such that

$$\bar{G} = (\bigcup_{h \in N_i} \bar{M}_h, \bar{A}) \tag{5.19}$$

where $\bigcup \bar{M}_h$ gives the set of nodes such that $\bar{M}_h = \{m \in M | pp_{i,n,m} > 0\}$ for $n = so_h|i$, and $\bar{A}$ gives the set of directed arcs such that $(m1,m2) \in \bar{A}$ if $\eta_{m1,m2}|n1 \rightarrow n2|i|j \geq 1$ (5.17). Then, any path in $\bar{G}$ whose ending nodes are a module from $\bar{M}_{h1}$ and a module from $\bar{M}_{h2}$, where $h1 = 1$ and $h2 = |N_i|$ is a feasible solution (i.e., a feasible processing route) to (5.12)-(5.17).

Let $\delta_{h,m2}$ denote the module to be visited by a part of type i for executing the operation $so_{h-1}|i$ provided that the **potential processing route** up to the execution level h for $h = 2,..,|N_i|$ and module $m2$ for $m2 \in \bar{M}_h$ (hereafter, $ppr(h,m2)$) is chosen. Note that $ppr(h,m2) = \{ppr(h-1, \delta_{h,m2}), m2\}$. Our shortest path based approach requires that $\delta_{h,m2}$ be the module such that

$$\delta_{h,m2} = arg.min \{ d_{m1,m2}|h \quad \forall m1 \in \bar{M}_{h-1} | (m1,m2) \in \bar{A} \} \tag{5.20}$$

where $d_{m1,m2}|h$ gives the total absolute deviation of the actual workload $\{\bar{al}_{h,m}\}$ from the expected workload $\{\bar{el}_{h,m}\}$ at the loading level j and execution level h, provided that the $ppr(h-1,m1)$ is chosen and the module $m2$ is selected for the level h. It can be expressed

$$d_{m1,m2}|h = \sum_{m \in M} | \ \bar{al}_m|h,m2 - \bar{el}_m|h,m2 \ | \tag{5.21}$$

where

$$\bar{al}_m|h,m2 = \bar{al}_m|h-1,m1 \ + \ \Delta \tag{5.22}$$

$\Delta = \bar{t}_{i,n,m}$ (5.3) (being $n = so_h|i$) for $m = m2$ and otherwise it is zero, and

$$\bar{el}_m|h,m2 = \frac{\bar{tl}_m}{\hat{tl}} \sum_{m \in M} \bar{al}_m|h,m2 \tag{5.23}$$

where $\bar{tl}_m$ and $\hat{tl}$ are given by (5.1) and (5.2), respectively.

Note that (5.22) for $h = 2$ requires $\bar{al}_m|1,m1$ for $\forall m \in M$; it is given by (5.24).

$$\sum_{g \in J_{j-1}} \sum_{n \in N_{lo(g)} | pr_{g,n} = m} \bar{t}_{lo(g),n,m} \ + \ \Delta' \tag{5.24}$$

where $\Delta' = \bar{t}_{i,n,m}$ (being $n = so_1|i$) for $m = m1$ where $m1 \in \bar{M}_1$ and, otherwise, it is zero.

The processing route $\{pr_{i,n}\}$ proposed by the algorithm is the $ppr(h,m2)$ for $h = |N_i|$, where $m2$ is the module from the set $\bar{M}_h$ with the smallest deviation (5.20) (note that it gives the value of $\bar{z}_i$).

2. **Obtaining the loading frequency**. Assume that $|J| = 4$. If after sequencing a given part, it is detected that the loading sequence e.g. x1 parts of type A, x2 parts of type B, x3 parts of type A and x4 parts of type C is repeated for a given number of 'cycles' then it is suggested to use this frequency for loading the parts that have not yet been loaded.

3. **Improving the loading sequence**. The improving is obtained by performing a new local optimization by introducing partial changes in the sequence. The changes to analyze consist of moving the part to be loaded at the j^{th} level to the g^{th} level, where $g = j+1,...j1$ and

$$j1 = min\{|J|, j + r\%|J|\} \tag{5.25}$$

for a given $r > 0$, such that the parts that are sequenced in between are advanced one level. Note that $j1$ gives the highest level for any partial change; the distance $j1 - j$ should balance the potential improvement with the required computing time (see computational experience reported below). The re-sequence only consists of re-ordering the loading of some parts in the FMS, without re-allocating the modules of the processing routes. Let $i1 = lo(j)$, $i2 = lo(j+1)$, $i3 = lo(g-1)$ and $i4 = lo(g)$. Then, the analysis of the potential re-

sequencing of a given part only needs one additional computation. It is the deviation of the actual workload from the expected workload at the level $g-1$, where now a part of type $i4$ is to be loaded; and the types of the parts already loaded are given by the set $\{lo(k)\ \forall k \in J_{g-1}\} \bigcup \{i2,...,i3\}$. Note that the deviation related to the level g (where a part of type $i1$ is to be loaded) does not change.

Final note. If the production volume is big enough, it is wise to partition it such that a 'part' is defined by a lot; obtaining the appropriate lot's size for each part type is up to now an open problem.

For illustrative purposes, let the case shown in Table 7; the results are given in Table 8; total deviation, 47.22.

Table 7. Operations execution sequences and routing proportions

Table 7a. Routing proportions for i=1

n1,n2	rpm1,m2					
	m1,m2	rp	m1,m2	rp	m1,m2	rp
1,2	2,1	0.33	2,3	0.17	4,4	0.50
2,3	1,2	0.33	3,2	0.17	4,4	0.50
3,4	–	–	2,2	0.50	4,4	0.50

Table 7b. Routing proportions for i=2

n1,n2	rpm1,m2					
	m1,m2	rp	m1,m2	rp	m1,m2	rp
1,2	3,3	0.25	3,4	0.50	4,4	0.25

Note: $|N_1| = 4$; $|N_2| = 2$; $D_1 = 6$; $D_2 = 4$; $t_{i,n} = n\ \forall n \in N_i, i \in I$; $\{e_{m,t} = 1\}$; $so_h|i = h$ for $h = 1,...,|N_i|, i \in I$

5.4. COMPUTATIONAL EXPERIENCE

The algorithm was tested by implementing a prototype running on the IBM 4381-P13 with VM/SP-4.0; it was written in PL/I and compiled with the version 1.3. Table 9 reports some computational experience. The description of P1 is given by Table 7; the results are shown in Table 8. P2 is related to the case shown in Table 5 and Figure 4. P3 to P9 are real-life problems.

Two types of improvements on the solution given by the Step 1 of the algorithm are performed for each problem. They are related to the values 20 and 100 of the parameter r in (5.25). The headings of the table are as follows. *aver* $|N|$ and *aver na* give the average number of operations and routing proportions per each part type, respectively. $T1$, CPU time (secs.) till the first solution is obtained. *nimp*, number of improvements. $\%imp = (z1 - z2)/z1\%$, where $z1$ and $z2$ give $\sum d_j\ \forall j \in J$ (5.8) related to the first solution and last improvement, respectively. T, total CPU time.

From the results shown in Table 9, it seems that an improvement on the shortest path based solution should be performed. But, higher values of r do not necessarily imply better improvements. Analyzing the potential improvement of all higher levels (case $r = 100$) of each loading level requires an additional computing time that is not balanced with the improvement achieved. $r = 20$ seems to balance both parameters; note that the maximum time required for solving any problem was 128 seconds and most of the problems were solved in less that 30 seconds.

Table 8. Parts loading sequence in FMS and processing route for the case shown in Table 7.

FMS loading sequence		Processing route				tl_j	d_j
		n=1	2	3	4		
j	lo(j)	m	m	m	m	(5.5)	(5.8)
1	2	3	4	–	–	3	2.33
2	1	2	1	2	2	13	9.89
3	1	4	4	4	4	23	2.47
4	2	3	3	–	–	26	4.06
5	2	4	4	–	–	29	3.33
6	1	2	1	2	2	39	10.08
7	1	4	4	4	4	49	2.56
8	2	3	4	–	–	52	2.78
9	1	2	3	2	2	62	9.72
10	1	4	4	4	4	72	0.00

n and m: operation to be executed and module to be visited in the processing route.

Final remarks

The special case $\overline{m} = 1$ (i.e., only one module may and must be assigned to each process) does not require to solve the model (5.9)-(5.16), since there is only one processing route per each part type. In any case, the proposed algorithm does not attempt to guarantee the solution's optimality, but it is as much as we can do.

Should the computing time not be a serious restriction and the dimensions of the problem not too-big, an attempt could be made for solving the model (5.10)-(5.17) till optimality. Note that the dimensions depend on the value of $\overline{m}$; from other point of view, higher the value of the loading level fewer available processing routes and, then, smaller is the model. Note that the processing route obtained at any loading level for a given type of part either is a feasible solution for the next level or it is the starting solution for obtaining the lower bound of the optimal one; in both cases, the algorithmic approach whose framework is given in Section 3 and Figure 3 could be applied. Implementing this alternative is not an easy task but, we believe, it is worthy to try it.

Table 9. Computational experience with the Parts loading algorithm

Prob#	|I|	|J|	aver |N|	aver na	TI	r=20			r=100		
						nimp	%imp	T	nimp	%imp	T
P1	2	10	3	6	0.01	1	3.6	0.01	4	8.8	0.02
P2	1	100	7	25	0.44	16	5.4	0.73	16	5.4	1.25
P3	5	83	15	40	2.30	47	11.0	2.60	49	11.6	3.00
P4	8	231	25	122	5.21	38	12.8	15.32	37	12.1	21.64
P5	10	210	32	138	8.55	96	7.7	12.97	111	7.8	16.84
P6	10	345	19	85	15.34	201	13.0	28.88	182	12.2	38.62
P7	8	838	28	107	21.75	350	5.4	31.80	428	6.2	56.50
P8	10	1432	31	124	56.38	428	16.0	71.08	455	15.8	96.88
P9	9	1620	43	172	99.07	358	9.0	128.61	358	9.0	181.89

6. CONCLUSIONS AND TOPICS FOR FUTURE RESEARCH

The (mathematical programming) models described in this paper presents an alternative for the production planning of a generic FMS floor shop problem. The output of the so-called **generative** system where the models are to be included is recommended to be used as input for an **evaluative** system (based on simulation); see Engelke et al. (1983). The latter does not provide any alternative for the production planning; but, by using deterministic and probabilistic data in a de-aggregated environment, it analyzes the performance and implications of the production planning provided by the generative system.

The main conclusions that can be drawn from this work are as follows:

1. Higher flexibility in the FMS, broader set of production planning alternatives to be simulated and greater difficulty on obtaining the best solution. It seems that a computerized system for narrowing the set of alternatives to evaluate should be investigated; this work is in the direction of this aim.

2. Given the complexity of the problem and our current algorithmic technology, a generic planning problem has to be decomposed in subproblems; then, the execution of the related models is hierarchical in nature. Models addressing broader problems must be executed first; their results are used by models with more specific short-term goals.

3. Based on our computational experience, we may conclude that the hierarchical approach presented in this paper can be a useful tool for providing good candidate solutions on realistic planning problems.

The application of OR to FMS is still in its infancy. Some open problems within the framework presented here are as follows:

1. A hierarchical categorization of a broader set of models.

2. Good heuristics (and special classes of expert-like systems) that use a graphic-interactive methodology are needed; the planner-system interaction may help to solve problems with some targets and constraints that are only in the mind of the planner.

3. Interfacing simulation with mathematical programming. Mathematical programming can be used without great difficulty to provide input for simulation models. To our knowledge, the other way around is still an open problem; we believe it is worthy to investigate how simulation may help to adding and/or deleting constraints in mathematical programming models.

4. High quality research is on going for identifying valid inequalities for (general and specific) pure 0-1 problems; many models for FMS production planning belong to this category. But, there is also a broad application field for mixed 0-1 problems that deserves to do so.

5. It could be interesting to exploit such special objective functions as minimizing 'bottlenecks' and absolute deviations for more general problems than the specific problems (such as LTP and AP) for which we have to-day the know-how. There is a variety of models for FMS production planning with these types of functions.

ACKNOWLEDGEMENT

This work strongly benefits from fruitful discussions with Christof Dillenberger, Helmut Engelke, Jens Grotrian, Rolf Krumholz, Claus Scheuing and Arno Schmackpfeffer specially on the conceptualization of some models, and the role of the graphic-interactive approach in FMS production planning.

REFERENCES

Ammons, J.C., C.B. Lofgren and L.F. McGinnis. 1984. A large-scale work station loading problem, in Stecke, K.E. and R. Suri (eds.) 249-255.

Abraham, C., B. Dietrich, S. Graves, W. Maxwell and C. Yano. 1985. A research agenda for models to plan and schedule manufacturing systems, report RC-11329, IBM Research, Yorktown Heights, NY.

Balas, E. and N. Christofides. 1981. A restricted Lagrangian approach to the Traveling Salesman Problem, Mathematical Programming 21, 19-46.

Berrada, E. and K.E. Stecke. 1983. A branch and bound approach for machine loading in Flexible Manufacturing Systems, working paper 329-b, Graduate School of Business Administration, University of Michigan, Ann Arbor, Michigan. See also Management Science 32 (1986) 1316-1334.

Carpaneto, C., S. Martello and P. Toth. 1985. Linear assignment problems, School of Combinatorial Optimization, Rio de Janeiro.

Chakravarty, E.K. and A. Shtub. 1984. Selecting parts and loading Flexible Manufacturing Systems, in Stecke, K.E and R. Suri (eds.) 284-289.

Chatterjee, A., M.A. Cohen, W.L. Maxwell and L.W. Miller. 1984. Manufacturing Flexibility: models and measurement, in Stecke, K.E. and R. Suri (eds.) 49-64.

Crowder, H., E.L. Johnson and M. Padberg. 1983. Solving large-scale zero-one linear programming problems, Operations Research 31, 803-834.

Engelke, E., J. Grotrian, C. Scheuing, A. Schmackpfeffer, W. Schwarz, B. Solf and J. Tomann. 1985. Integrated Manufacturing Modeling System, IBM J. of Research and Development 29, 343-355.

Escudero, L.F. 1979. Special sets in Mathematical Programming, in Alarcon, E. and C. Brebbia (eds.), Applied Numerical Modeling (Pentech Press, London) 535-551.

Escudero, L.F. 1986. An integrated approach for generating production planning models in FMS, working paper, IBM GMTC, Sindelfingen, FRG.

Escudero, L.F. 1987. An (inexact) algorithm for the Sequential Ordering Problem, European J. of Operational Research (submitted for publication).

Freville, A. and G. Plateau. 1986. Heuristics and reduction methods for multiple constraints 0-1 linear programming problems, European J. of Operational Research 24, 206-215.

Greene, T.J. and R.P. Sadowski. 1986. A mixed integer program for loading and scheduling multiple flexible manufacturing cells, European J. of Operational Research 24, 379-386

Grotschel, M., M. Junger and G. Reinelt. 1984. A cutting plane algorithm for the linear ordering problem, Operations Research 34, 1195-1220.

Guignard, M. and K. Spielberg. 1977. Propagation, penalty improvement and use of logical inequalities, Mathematics of Operations Research 25, 157-171.

Gunn, T.G. 1982. The mechanization of design and manufacturing, Scientific American 247, 870-108.

Haines, C.L. 1985. An algorithm for carrier routing in a flexible material-handling system, IBM J. of Research and Development 29, 356-362.

Hoffman, K. and M. Padberg. 1986. LP-based combinatorial problem solving, Annals of Operations Research 5, 145-194.

IBM. 1975. MPSX/370, Mathematical Programming System Extended/370 and MIP/370, Mixed Integer Programming/370; Ref. manuals SH19-1095 and SH19-1099 . Introduction to the Extended Control Language, 1978, Ref. manual SH19-1147). Newletter, 1979, SN19-1132.

Johnson, E.L., M.M. Kostreva and U. Suhl. 1985. Solving 0-1 integer programming problems arising from large-scale planning models, Operations Research 33, 803-819.

Karp, R.M. and J.M. Steele. 1985. Probabilistic analysis of heuristics, in Lawler, E.L., J.K. Lenstra, A.H.G. Rinnooy-Kan and B. Shmoys (eds.), The Traveling Salesman Problem, A guided tour of combinatorial optimization, Wiley, NY, 181-205.

Kusiak, A. 1985. Integer programming approach to the process planning problem, Intern. J. of Advanced Manufacturing Technology 1, 73-83.

Kusiak, A. and G. Finke. 1985. Modeling and solving the flexible forging module scheduling problem, working paper 85/06, Dept. of Mechanical and Industrial Engineering, University of Manitoba, Winnipeg, Canada.

Martello, S., 1983. An enumerative algorithm for finding Hamiltonian circuits in a directed graph, ACM Transactions on Mathematical Software 9 (1983) 131-138.

Sandi, C. 1975. Solution of the machine loading problem with binary variables, in B. Roy (ed.), Combinatorial Programming: methods and applications, D. Reidel, Dordrecht-Holland, 371-378.

Stecke, K.E. 1983. Formulation and solution of nonlinear integer production planning problems for flexible manufacturing systems, Management Sciences 29, 273-288.

Stecke, K.E. 1986. A hierarchical approach to solving machine grouping and loading problems of flexible manufacturing systems, European J. of Operational Research 24, 369-378.

Stecke, K.E. and T.L. Morin. 1985. Optimality of balanced workload in Flexible Manufacturing Systems, European J. of Operational Research 20, 68-82.

Stecke, K.E. and R. Suri (eds.). 1984. Flexible Manufacturing Systems: Operations Research Models and Applications, University of Michigan, Ann Arbor.

Stecke, K.E. and F.B. Talbot. 1983. Heuristic loading algorithms for flexible manufacturing systems, Proceedings of the 7th Intern. Conf. on Production Research, Windsor, Ontario.

Van Roy, T.J. and L.A. Wolsey. 1987. Solving mixed integer programming problems using automatic reformulation , Operations Research 35, 45-57..

Whitney, C.K. and T.S Gaul. 1984. Sequential decision procedures for batching and balancing in FMSs, in Stecke, K.E. and R. Suri (eds.) 243-248.

Wittrock, R.J. 1985. Scheduling algorithms for flexible flow lines, IBM J. of Research and Development 29, 401-412

Wittrock, R.J. 1986. An adaptable scheduling algorithm for flexible flow lines, report RC-11387 IBM Research Division, Yorktown Heights, N.Y.