Production Systems and Information Engineering
Volume 3 (2006), pp. 121-137

MODELING AND SOLVING OF THE EXTENDED FLEXIBLE FLOW SHOP SCHEDULING PROBLEM

GYULA KULCSÁR
University of Miskolc, Hungary
Department of Information Engineering
kulcsar@ait.iit.uni-miskolc.hu

FERENC ERDÉLYI
Production Information Engineering Research Team (PIERT) of the
Hungarian Academy of Sciences
University of Miskolc, Hungary
Department of Information Engineering
erdelyi@ait.iit.uni-miskolc.hu

[Received October 2005 and accepted April 2006]

Abstract. This paper discusses the production control problems of customized mass production, which can be described as combining make to stock and make to order type production at the same time. In order to solve scheduling and resources allocation issues, a new computer model for customized mass production will be presented. The focus has been set to determine alternative routes and machines allocation for feasible scheduling. We have developed a computer framework to check different approximate heuristic algorithms, the result of which will be summarized in this paper.

Keywords: shop floor scheduling (assigning, sequencing, simulation), alternative routes, parallel machining, constraint, due date

1. Introduction

In essence there are two kinds of manufacturing: make to order (MTO) and make to stock (MTS). The production of unique or complex goods falls into the first category. MTO production can be characterized by individual or customer specified products usually designed from a set of firm level components, and small batch production on universal machines at job-shop environment.

Make to stock manufacturing is used in mass production, where the finished products are delivered from a warehouse when customer requests and purchases them. Mass production technology usually has automated manufacturing and/or assembly lines, highly skilled or specialized workers, big lot sizes, automated quality checking, automated packing operations, and relatively high material stock level.

Certain manufacturers may use both procedures: in addition to satisfying their accepted orders they can utilize their manufacturing resources for producing goods for stock. In this customized mass production paradigm the firms plan their production partially for external direct orders, arriving from logistic or shopping centers but to reach better delivery dates they must make forecast for manufacturing semi-finished products and buying materials with long external lead time [5]. In this business environment there are lots of uncertainties originated from incoming orders and unavailability of machines, equipment or human resources. For this reason, real time production data collection, fast interactive information management and effective scheduling of tasks are the most important tools to achieve production goals. These are the main functions of Manufacturing Execution Systems [7].

2. Scheduling Problems in Customized Mass Production

2.1. Classification of Shop Floor Scheduling Models

Scheduling is the allocation of a set of well-defined resources to a set of given tasks subject to some pre-determined constraints, in order to satisfy a specific objective. In order to formulate a scheduling problem, the specification $\alpha/\beta/\gamma$ is typically used [1], where:

- α machine environment,
- β processing characteristics and constraints, and
- γ objective functions.

Production of parts is carried out in batches. The batch size may vary from one part (manufactured in job shops) to millions of parts (manufactured in production lines). Depending on how the jobs are executed at the shop floor (i.e. the sequence, in which jobs visit machines), we can classify manufacturing systems as one of:

- serial systems (also called flow shops) or
- non-serial systems (job shops).

The following figures (Figure 1 and Figure 2) summarize some typical flow types.

Figure 1. Flow Shop Scheme

In flow shop structure, there are machines *(m 1,... M)* in sequence. Unlimited intermediate storage between two successive machines is usually assumed. All jobs have the same routing. Each job has to be processed on each one of the *m* machines. Permutation flow shop means that the queues in front of each machine operate according to FIFO (First In First Out discipline).

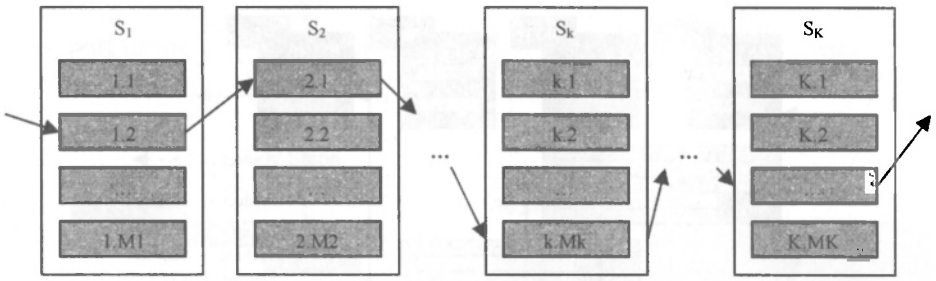

Figure 2. Flexible Flow Shop Scheme

The flexible flow shop environment has stages, at stage S_k *(k=1, ..., K)* there are M_k identical machines in parallel. There is usually unlimited intermediate storage between two successive stages. Each job has to be processed at each stage on any of the machines.

In this paper, we are focusing on extended flexible flow shop machine environments with alternative routes, parallel machines and unlimited intermediate storages. The following sections will deal with an extended flow shop environment with four stages (see Figure 3).

2.2. Extended Flexible Flow Shop Scheduling Problem

The problem is inspired by a real case study concerning a Hungarian firm specialized in lighting products. It is a customized mass production approach so that an order-book for a given time period corresponds to different products to be produced in required quantity. We concentrate on generating short-term production schedule of the manufacturing processes.

This section covers the scheduling problem in detail, the related entities of the system, and how they relate to each other. The whole scheduling model can be described as follows. We present the machine environment (α), the processing characteristics and constraints (β) and the objective functions (γ).

Product Type: The product type can be defined as the combination of components that a machine is capable of handling. The combination uses the AND operator and the OR operator to combine various lists of components.

Production Orders: There are production orders. A production order includes the type of the final product to be manufactured, the required quantity and the defined due date. In order to satisfy a production order, the components are taken through various processes before finally becoming the final product. Manufacturing includes a set of steps that involve the actual production of the final product.

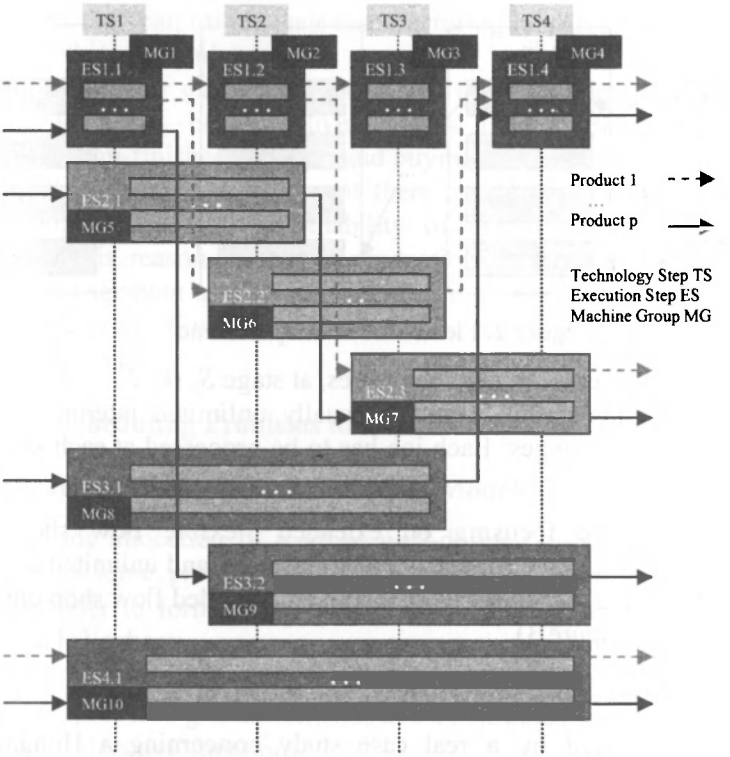

Figure 3. Extended Flexible Flow Shop Scheme

Technology Steps: The steps under the technology processes are termed as technology steps. Typically, in manufacturing we have four technology steps like preparation, assembly, quality checking and packaging. Preparation is the first step of the technology processes when defined properties of certain components have to be modified. Assembly is the technology step in which the components are assembled together, quality checking has two parts: a forced wait time when the quality of the product is observed and ascertained before going through packaging finally. A technology step may include some operations, but we suppose that no pre-emption is allowed at the level of the technology steps.

Execution Steps, Execution Routes: There is a very important concept namely execution step in this environment. The execution step is a well-defined set and sequence of technology steps (Figure 4). It describes which technology steps to be processed on the same production line. If the execution step includes two technology steps (i.e.: TS1 and TS4), it has to include all technology steps, which are between these two steps (TS1, TS2, TS3, TS4), as well. Moreover, the sequence of execution steps is called execution route. So each execution route

includes all technology steps required in order that the final product can be produced. An execution route can include one or more execution steps, but the common part of included execution steps, which is a set of technology steps, has to be an empty set (Figure 5).

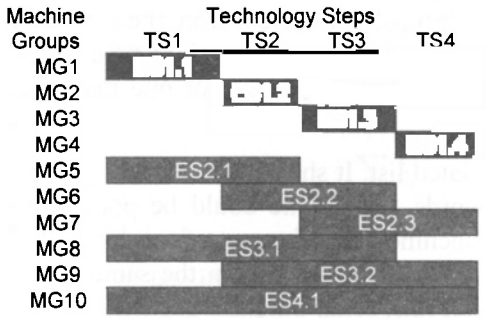

Figure 4. Execution Steps

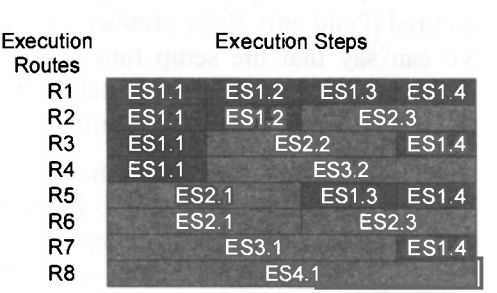

Figure 5. Execution Routes

Pallets: Each product is normally packaged on standard sized pallets. Each pallet consists of a pre-defined number of the finished products. Even though the physical pallets come into existence only after packaging is over, for convenience reasons, we start looking at the logical pallets right from the beginning. Hence, we schedule pallets (job). A production order is first identified to be consisting of a particular number of pallets and the production order will be closed when all of these pallets have gone through all the technology steps.

Machines: In our scheduling model, a machine is a production line that consists of a group of workplaces, which are lined in a sequence; the output of the first workplace becomes the input of the second one and so on. Typically these production lines are inseparable unites. Hence when it comes to scheduling, then the production line should be considered as one unit and not the individual workplace.

Production Rate and Machine Capacity: During manufacturing, we have to schedule units (pallets) on production lines (machines). In order to estimate the process time taken by a pallet on a machine, we need to know the production rate and the capacity of that machine to process that pallet. Production rate is normally specified as quantities producible per time unit on that machine. The capacity of a machine could vary based upon what is the final product it is producing and what is the effective shift time in the calendar.

Process Time: The production rate of the machine is required while computing the process time of a pallet on the machine. Process Time means the processing time of a pallet spending on a machine (in time unites).

$$\text{Process Time} = \text{Scheduled Quantity} / \text{Production Rate}. \quad (1)$$

Setup Time: This is one of the most important properties of the machine (production line). This does not affect the producibility of a final product directly. By definition, a setup time (changeover time) gives the time delay to changeover from one product type to another product type. In our current model, the setup is required if and only if the product type of the last pallet different from the next one. We can say that the setup time value only depends on the product type to be processed, so it is allowed to define multiple value setup time for one machine. Setup time is specified in time units.

Machine Group: Each machine has an associated list. It shows all technology steps can be executed on a machine. In other words, a machine could be potentially capable of executing an execution step. A machine group is a set of machines that can execute the same execution step (Figure 4). The machines in the same group are parallel machines with different production rate, setup time and capacity.

Alternative Execution Routes: A given final product can be produced differently, because there are different execution routes on which the required components are taken through before becoming the final product. These alternative routes differ in the execution steps. In addition, each execution route may include parallel machines assigned to one or more execution step. In our model, there is a dynamic list which describes the available execution routes at a given time period for each final product.

Component availability: In this issue we use a simplification of the original problem. It means that we do not focus on all components availability; instead we suppose all of the required material available in the needed quantity from the CST (Constrained Start Time) of the pallets on the machine. CST of the pallet specifies the earliest time when the first execution step of the pallet can start from the aspect of the component (material) availability.

Objectives: A scheduling objective is a measure to evaluate the quality of a certain schedule. In real-life situations, there are many (delivery capability, machine utilization rate, stock or WIP level, they are usually conflicting) objectives. For delivery capability, one can distinguish two types of objectives:

- due date related objectives and
- non due date related objectives.

For due date related objectives, we assume that there are jobs J_i $(i=1,...,N_J)$. Each job J_i has due date d_i and release date r_i. The due date represents the commitment of the company with a customer. The release date implies the availability of components from the beginning. We denote the finishing time of job J_i by C_i. The following definitions may be defined for each job:

Lateness of a job: $L_i = C_i - d_i$. (2)

Tardiness of a job: $T_i = \max(0, L_i)$ (3)

Earliness of a job: $E_i = \max(0, -L_i)$ (4)

With each of these functions F_i we get some possible objectives. So the most important objectives may be as follows:

Maximum: $\gamma = \max(F_i)$. (5)

Total: $\gamma = \sum_i F_i$ (6)

Average: $\gamma = \frac{\sum_i F_i}{n}$ (7)

Number of late jobs: $\gamma = |\{i \mid T_i > 0\}|$. (8)

Usually, not all of the jobs are equally important. Weights w_i can be assigned to each job representing the relative importance of the jobs. Some measures that take into account the different weight of the jobs are as follows:

Weighted maximum: $\gamma = \max(w_i F_i)$. (9)

Weighted total: $\gamma = \sum_i w_i F_i$ (10)

Weighted average: $\gamma = \frac{\sum_i w_i F_i}{n}$ (11)

The most common objective functions, which are non due date related, are as follows:

Makespan: $\gamma = \max(C_i)$. (12)

Total flow time: $\gamma = \sum_i C_i$ (13)

Weighted total flow time: $\gamma = \sum_i w_i C_i$ (14)

It is well known, that the optimal solution can be quite different if the chosen objective changes. Depending on the fixed objectives, each decision maker wants to minimize a given criterion. On one hand, the commercial manager is interested in satisfying orders by minimizing the lateness. On the other hand the production manager wishes to minimize the work in process by minimizing the maximum flow time.

3. Solution of the Scheduling Problem

In this section we outline our approach to solve problem described in section 2.2. We show the developed data model and the basic steps of our methods. Then we present a computer application of this solution.

3.1. Basic Data Structures

In our model, we use indexed arrays in order to accelerate the calculation. In these arrays, there are no full length identifiers and attributes of entities (i.e.: jobs, machines, routes and so on), instead there are indexes, which are non-negative integer values assigned to the entities, to point to the position of the target object in the base array. Therefore, in any of indexes of a given array, we can use any value of the same array or another array. In order to indicate an element of one-dimensional or two-dimensional array, we use the following formulations:

- ARRAY_NAME[ROW_INDEX]
- ARRAY_NAME[ROW_INDEX][COLUMN_INDEX]

If an array element is a data structure made up of fields, we use the dot operator to refer to a specified data field by using the field name:

- ARRAY_NAME[ROW_INDEX].FIELD_NAME
- ARRAY_NAME[ROW_INDEX][COLUMN_INDEX].FIELD_NAME

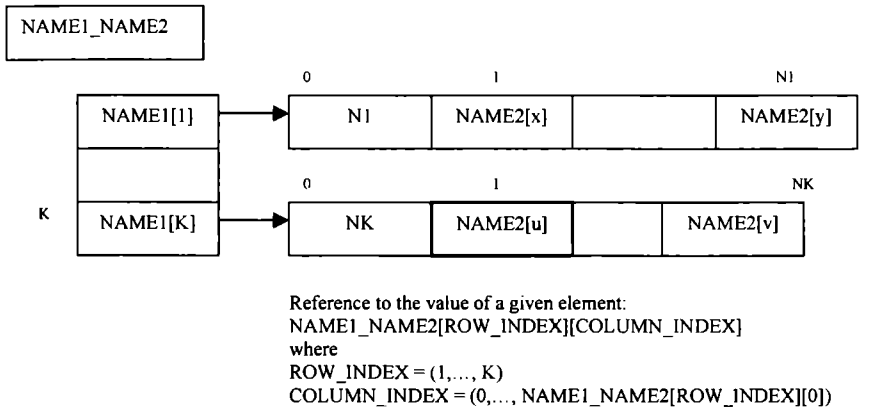

Figure 6. General Structure of two-dimensional arrays

General structure of two-dimensional arrays with variable number of row elements can be seen on Figure 6. It shows the association of two different type arrays (NAME1 and NAME2). The actual array can be specialized from the general structure in such a way that basic array corresponds to NAME1 and related array corresponds to NAME2.

3.2. Data Model

Using the above formulations, we can describe the entities of the scheduling model detailed in section 2.2 as follows:

In the system, there are different final products p (p $1,\ldots, N_P$) which may be produced. There is an order book for a given time period. It has production orders o (o $1,\ldots, N_O$). Each production order o includes the type of the final product $O[o].P$, the required quantity $O[o].Q$ and the defined end time $O[o].ET$ (due date).

At the shop floor, pallets can be moved. Each pallet consists of a pre-determined number $NP[p]$ ($p = 1,\ldots, N_P$) of the finished products p. Each production order o is identified to be consisting of a particular number of pallets. We schedule pallets, one pallet means one job. So we have jobs i ($i = 1,\ldots, N_J$) altogether. Each job has four attributes: $J[i].P$ means the final product p, $J[i].Q$ means the quantity of the products, $J[i].CST$ means the constrained start time and $J[i].CET$ means the constrained end time.

Each job i has to visit four technology steps $TS[t]$ ($t=1,\ldots, 4$) in the same sequence. The workshop contains ten possible machine groups mg ($mg = 1,\ldots, 10$) connected to each other in a given configuration (Figure 3.). Each machine group mg contains a pre-defined number of machines. There is a two-dimensional array named MG_M which describes the list of machine groups with the machines which belong to them. The structure of MG_M is inherited from the general structure shown on Figure 6. Machine group corresponds to NAME1 and machine corresponds to NAME2.

In a given machine group mg, each machine can process the same execution step which is one of the well defined execution steps es (es $1,\ldots, 10$). We have machines m ($m=1,\ldots, N_M$) altogether. Each machine m may have N_P different production rate $M\_PR[m][p]]$ ($m =1,\ldots, N_M$ and p $1,\ldots, N_P$). Similarly, each machine m may have N_P different setup time $M\_ST[m][p]$ ($m =1,\ldots, N_M$ and $p = 1,\ldots, N_P$), according to the definitions of the setup time and production rate in section 2.2.

Jobs can be moved on eight possible execution routes r ($r = 1,\ldots, 8$) (see Figure 5.). Each route r includes a sequence of machine groups. These assignments are defined in an array named R_MG, which is a specialization of the structure shown on Figure. 6. Machine group corresponds to NAME1 and machine group corresponds to NAME2.

In our model, utilizing what has gone before, we can determine an array P_R which describes the available execution routes in the actual time period for each final product p. The array P_R can also be specialized from the general structure in such

a way that final product corresponds to NAME1 and execution route corresponds to NAME2.

We suppose the shop floor has already been loaded, so the actual state of the system has to be known in order to calculate start time and end time of each job on each assigned machine. It means that the effect of the last confirmed schedule has to be available. These data can be obtained from array *M_ENGAGED* which shows the earliest time of each machine when the machine is available, *M_ENGAGED[m]* ($m = 1,..., N_M$).

Additional arrays have been defined to store the result of the scheduling. There is a special array named *J_A* which includes the route and machines assigned to jobs in the following way: *J_A[i][am]* ($i = 1,..., N_J$ *and* $am = 0,...,$ *R_MG[J_A[i][0]][0]*). Where:

- *i* means a job,
- *J_A[i][0]* means the assigned route,
- *R_MG[J_A[i][0]][0]* means the number of machines in the assigned route,
- *J_A[i][am]* ($am=1,...,$ *R_MG[J_A[i][0]][0]*) means the sequence of assigned machines.

There is an array named *MWLOAD* which shows the sequence of jobs on machines. This structure *MWLOAD[m][ai]* ($m= 1,...,N_M$ *and* ai $1,...,$*MWLOAD[m][0]*) is a specialization of the general structure (Figure 6.). In this case, machine corresponds to NAME1 and job corresponds to NAME2.

- *m* means a machine,
- *MWLOAD[m][0]* means the number of jobs on machine *m*,
- *MWLOAD[m][ai]* ($ai = 1,...,$ *MWLOAD[m][0]*) means the sequence of jobs to be processed on machine *m*.

Finally, we have defined an array named *MSTET* which stores the calculated times, which are as follows: *ST* start time, *SetT* setup time, *PT* process time and *ET* end time of the jobs *MWLOAD[m][ai]* ($ai \quad 1,...,$ *MWLOAD[m][0]*) on each machine *m* ($m = 1,...,N_M$).

3.4. Calculation Model

The calculation means numerical simulation of the production to calculate the time data of the operations. Inputs are jobs *i*, machines *m*, their assignments *J_A*, sequences of jobs on machines *MWLOAD*, abilities of machines *M_PR*, *M_ST* and availabilities of machines *M_ENGAGED*. Simulation of job *i* on an intermediate machine requires, among other things, the end time of job *i* on the previous machine and the shop floor environment has lots of junctions of the possible routes.

So we have to define the machine group sequence in which the calculation can be performed.

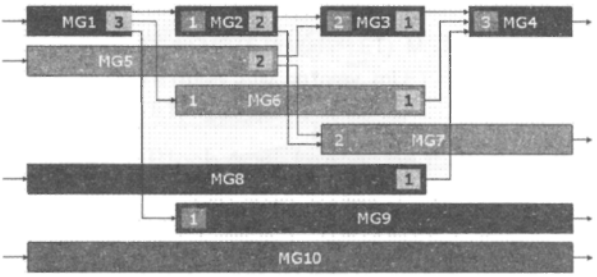

Figure 7. Precedence constraints between machine groups

Precedence constraints can be presented as a simple directed graph (Figure 7), in which the vertices are the machine groups and the edges show the required sequences of machine groups. For each machine group (MG), indegree and outdegree can be defined. Indegree means the number of inward directed edges from a given vertex and outdegree means the number of outward directed edges from a given vertex. At each machine group, the maximum of indegree and outdegree shows the number of possible routes which can include the given machine group. Possible sequence of the machine groups have been obtained by using the non-increasing indegree order of the machine groups.

$$D_{IN}(4) > D_{IN}(3) > D_{IN}(7) > D_{IN}(2) > D_{IN}(6) > D_{IN}(9) > D_{IN}(1) > D_{IN}(5) > D_{IN}(8) > D_{IN}(10). \quad (15)$$

The sequence is fixed in *PRI_MG* array which includes the priority of each machine group. The priorities are as follows:

Priority:	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
Machine Group:	{4, 3, 7, 2, 6, 9, 1, 5, 8, 10}

The main steps of calculation method can be seen on the flow chart (Figure 8.). The method goes in non-increasing priority order of the machine groups, and it calculates time data *(ST, SetT, ProcT, ET)* of jobs on each machine which is in the machine group. Start time data of jobs play the main role in calculation. This value of a given job *i* on an assigned machine *m* is determined by the following values:

- the constraint start time of the job *(J[i].CST)*,
- end time of the job on the previous machine *(MSTET[prev_m][i_on_prev_m].ET)*,
- end time of the previous job on the machine *(MSTET[m][ai -1])*,
- setup time of the job on the machine *(M_ST[m][J[i].P])* and
- the availability of the machine *(M_ENGAGED[m])*.

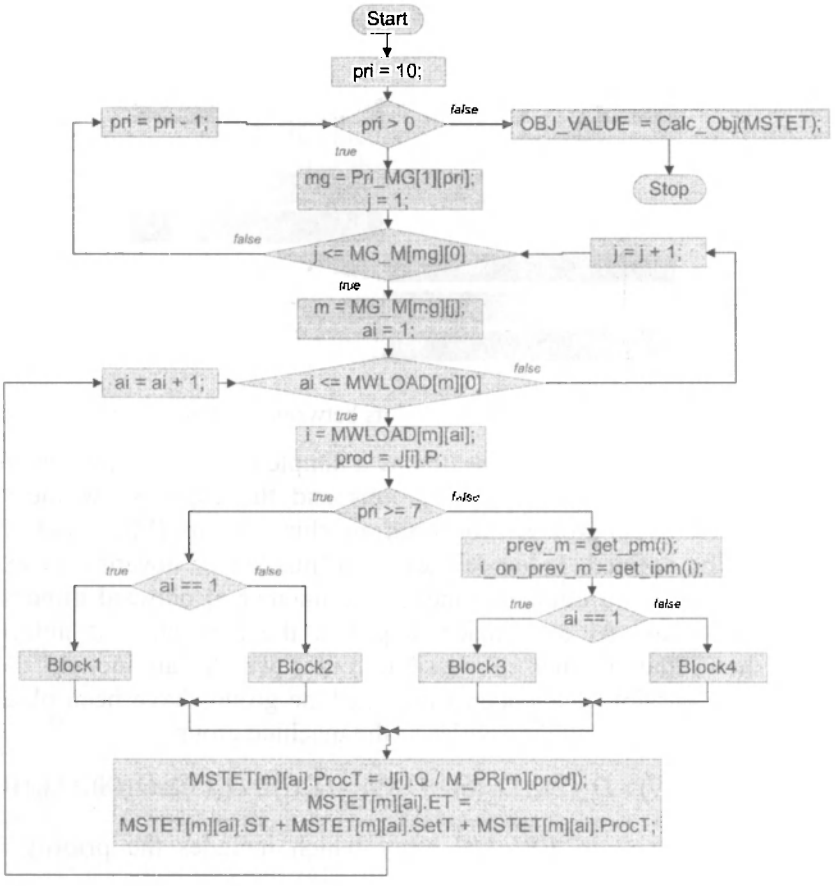

Figure 8. Flow Chart of Simulation

In point of view of start time and setup time four cases can be distinguished:

Block1: first job on first machine.

```
MSTET[m][ai].SetT = M_ST[m][prod];
MSTET[m][ai].ST = max( J[i].CST, M_ENGAGED[m] );
```

Block2: non first job on first machine.

```
if ( prod != J[MWLOAD[m][ai -1]].P ) MSTET[m][ai].SetT = M_ST[m][prod];
else   MSTET[m][ai].SetT = 0;
MSTET[m][ai].ST = max( MSTET[m][ai -1].ET, J[i].CST );
```

Block3: first job on non first machine.

```
MSTET[m][ai].SetT = M_ST[m][prod];
MSTET[m][ai].ST = max( MSTET[prev_m][i_on_prev_m].ET - MSTET[m][ai].SetT,
                       M_ENGAGED[m] );
```

Block4: non first job on non first machine.

```
if ( prod != J[MWLOAD[m][ai -1]].P )  MSTET[m][ai].SetT = M_ST[m][prod];
else   MSTET[m][ai].SetT = 0;
if ( MSTET[prev_m][i_on_prev_m].ET <= MSTET[m][ai -1].ET)  MSTET[m][ai].ST = MSTET[m][ai -1].ET;
else  MSTET[m][ai].ST = max( MSTET[prev_m][i_on_prev_m].ET - MSTET[m][ai].SetT,
                             MSTET[m][ai -1].ET);
```

Typical examles of working Block 4 can be seen on the Figure 9.

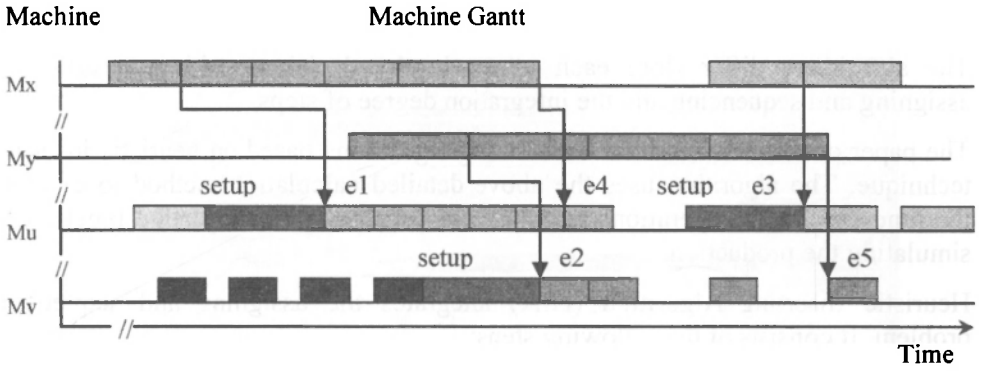

Figure 9. Working Block4

The outputs of the time calculation are *MSTET* array, which includes fixed times, and *OBJ_VALUE*, which stores the evaluated value of the chosen objective function.

3.5. Heuristic Algorithms

Our scheduling problem is difficult to solve because of its combinatorial nature. In order to define a schedule for the production of each job, it is necessary for each job i ($i = 1,...,N_J$):

1. to assign to one of the possible routes: $J\_A[i][0] = P\_R[J[i]\ P][ar]$ ($ar \in \{1,..., P\_R[J[i]\ P][0]\}$),
2. to assign to one of the possible machines at each possible machine group according to selected route: $J\_A[i][amg] = MG\_M[amg][am]$ ($amg = 1,..., R\_MG[J\_A[i][0]][0]$ and $am \in \{1, ..., MG\_M[amg][0]\}$),
3. to fix its position in the queue of each selected machine: $MWLOAD[J\_A[i][amg]][ai]$ ($amg = 1,..., J\_A[i][0]$ and $ai = 1,..., MWLOAD[J\_A[i][amg]][0]$),
4. to fix its starting time on each selected machine. $MSTET[J\_A[i][amg]][ai]$ ($amg = 1,..., J\_A[i][0]$ and $ai = 1,..., MWLOAD[J\_A[i][amg]][0]$).

Different heuristic procedures have been developed to solve the problem. These procedures are integrated into the scheduling engine (SE). At present, SE includes two kinds of classes of heuristic algorithms, which are as follows:

- Constructive algorithms,
- Iterative improvement algorithms.

The basic approach of our heuristic algorithms consists of tree steps:

1. Assigning: SE creates the *J_A*.
2. Sequencing: SE creates the *MWLOAD*.
3. Simulation: SE calculates the *MSTET*.

The algorithms differ from each others in the decision making in issues of assigning and sequencing and the integration degree of steps.

The paper outlines one of these scheduling algorithms based on heuristic insertion technique. The algorithm uses the above detailed calculation method to evaluate the time data of the operations and the value of the chosen objective function by simulating the production.

Heuristic Inserting Algorithm (HIA) integrates the assigning and sequencing problem. It consists of the following steps:

Step 1: Order the sequence of production orders according to the Earliest Due Date (EDD) rule, then in this sequence let the secondary consideration be the number of jobs of the production orders. Create the list of the jobs.

Step 2: Get the next job from the list.

Step 3: Enumerate all possible routes and machines for the job. Assign the job to each possible route in increasing order of the number of machine groups. Assign the jobs to each possible combination of available machines at each machine group of the route in decreasing order of production rates.

Step 4: On each first machine, insert the job into each position which is between two different batches. (On a machine, a batch means a sequence of jobs which is related to the same production order.)

Step 5: On the further machines of the route, the jobs flow through the system in order of arrival (First In First Out, FIFO).

Step 6: In each case use the calculation method.

Step 7: After simulation select the best solution according to the chosen objective function.

Step 8: Update data. If the list is empty, the algorithm is finished else go to Step 2.

3.6. Computer Application

We developed a computer application which consists of a problem generator, production simulator, scheduling engine and a database system. The main goal of this framework is that it can be an extremely useful tool supporting future studies of alternative scheduling algorithms.

The application uses sample data sets created by problem generator. The generator produces random problem instances with sizes and characteristics specified by user and then it writes them into the database. The generated data are well-defined random values, but the user can directly change certain data.

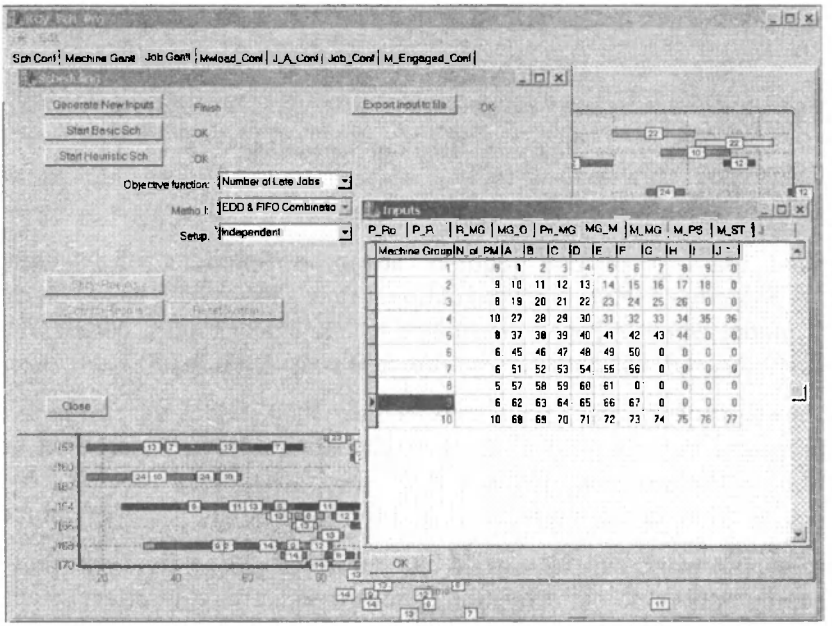

Figure 10. User Interface of Scheduler Engine

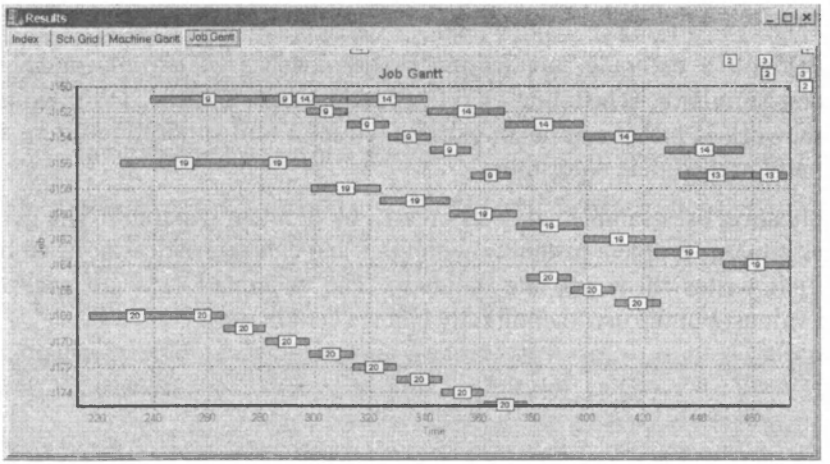

Figure 11. Result of Scheduling

4. Conclusions

Customized mass production requires advanced functions of Manufacturing Execution Systems (MES). The conventional flow shop model has to be extended to a new model which supports alternative technological routes, parallel machines and where setup and job characteristics are also considered.

In this paper, some possible extensions of flow shop model for customized mass production have been described. A new scheduling approach based on heuristic methods to solve extended flexible flow shop scheduling problems has been introduced. A computer program developed for this problem has been outlined.

Future research work will be carried out on investigating heuristic procedures, which can be applied to our scheduling problem and studying effect of change in the machine environment.

Acknowledgements

The research and development summarized in this paper was partially supported by the Hungarian Academy of Sciences (HAS) within the framework of Production Information Engineering Research Team (PIERT) established at the Department of Information Engineering of the University of Miskolc (Grant No. MTA-TKI 06108). The results are also connected with the NODT project entitled "VITAL" (National Office for Development and Technology founded by the Hungarian Government, Grant No.: 2/010/2004, project leader: László Monostori DSc). The authors would like to express their thanks for the financial support.

REFERENCES

[1] BRUCKER, P.: *Scheduling Algorithms*, Springer-Verlag, Berlin, 1998.

[2] KIS, T., ERDŐS, G., MÁRKUS, A., VÁNCZA, J.: *A Project-Oriented Decision Support System for Production Planning in Make-to-Order Manufacturing*, ERCIM News, No. 58, pp. 66-67, 2004.

[3] KOVÁCS, A., VÁNCZA, J.: *Completable Partial Solutions in Constraint Programming and Constraint-based Scheduling*, Principles and Practice of Constraint Programming, Springer LNCS 3258, pp. 332-346, 2004.

[4] KOVÁCS, L.: *The Methodology of Data Base Design and Management*, ComputerBooks, Budapest, 460 p. (In Hungarian), 2004.

[5] KULCSÁR, GY., HORNYÁK, O., ERDÉLYI, F.: *Shop Floor Decision Supporting and MES Functions in Customized Mass Production,* Conference on Manufacturing Systems Development - Industry Expectations, Wroclaw, Poland, pp. 138 – 152, 2005.

[6] KURNAZ, A., COHN, Y., KOREN, Y.: *A Framework for Evaluating Production Policies to Improve Customer Responsiveness*, CIRP Annals, Volume 54/1, pp. 401-406, 2005.

[7] MCKAY, K., N. WIERS, V., C.: *Unifying the Theory and Practice of Production Scheduling*, Journal of Manufacturing Systems, Vol. 18, No. 4, pp. 241-248, 1999.

[8] ZWEBEN, M., FOX, M.,S.: *Intelligent Scheduling*, Morgan Kaufmann Publishing, San Francisco, 1994.