Production Systems and Information Engineering
Volume 6 (2013), pp. 41-56

DETAILED PRODUCTION SCHEDULING BASED ON MULTI-OBJECTIVE SEARCH AND SIMULATION

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

MÓNIKA KULCSÁRNÉ FORRAI
University of Miskolc, Hungary
Department of Automation and Communication Technology
kulcsfm@mazsola.iit.uni-miskolc.hu

[Received January 2012 and accepted September 2012]

Abstract. This paper presents an advanced approach to solving fine scheduling problems of production in practice. It focuses on creating near-optimal feasible schedules considering detailed constraints and capabilities of the resource environment. It is a very important and complicated task to make an efficient schedule for the shop floor to include different types of facilities and operations. The proposed model supports the flexible usage of production goals and requirements simultaneously. The elaborated approach uses a special searching technique with multiple neighbouring operators and problem space transformation based on execution-driven simulation. Successful applications of the methods in real manufacturing environments are also demonstrated.

Keywords: scheduling, simulation, multi-objective optimization, production

1. Introduction

In modern manufacturing/assembling production environments, a great number of scheduling problems may occur. Production scheduling can be defined as the allocation of available production resources over time to perform a collection of tasks [1]. It is a very important decision making process at the operation level. Most of the scheduling problems are highly complicated and hard to solve owing to the complex nature of the applied model. Today, production engineering and management utilize more and more computer integrated application systems to support decision making.

This paper is primarily concerned with industrial scheduling problems, which require advanced scheduling software to assign limited available resources to the operation of jobs and to sequence the assigned operations on each resource over time. It is mainly concerned with discrete manufacturing, in which typically series

of items are produced. The series (batch or lot) can include very different numbers of pieces, from a single product (i.e. special part, complex or unique equipment) to thousands or millions of the same product (simple parts). In discrete manufacturing operations are executed on discrete, separate machines and workplaces. Depending on the arrangement of machines, robots, buffers and material handling devices, manufacturing systems may be characterized by different layouts (i.e. single machine, parallel machines, line, flexible line, group, etc.). In essence the execution of the operations requires the exact prior definition of the feasible routing alternatives [4].

2. Scope of research

The paper focuses on the fine (or detailed) scheduling function of Manufacturing Execution Systems. The main purpose of fine scheduling is to initiate a detailed schedule so as to meet the master plan defined at the Enterprise Resource Planning level. The scheduler is able to get the actual data of dependent production orders, products, resource environment and other technological constraints (tools, operations, buffers, material handling, etc.). The shop floor management configures the actual production goals and their priorities. Obviously, the management may declare various goals time by time. The scheduler has to provide a feasible schedule which meets the management's goals. The result of the scheduling process is a detailed production program which declares the releasing sequence of the jobs and the operations, assigns all the necessary resources to them and proposes the starting time of activities. The execution of the production program has to meet the predefined goals without breaking any of the hard constraints. The computation time of the solving process is also an important issue, especially with a large number of internal orders, jobs, operations, resources, technological variants and constraints.

In the literature, different flexible scheduling functions with various models are found. One of the main groups of these models is the flexible flow shop (FFS) scheme. A detailed survey of the FFS problem is given by Quadt and Kuhn [7], and Wang [10]. The FFS environment consists of stages that represent the fundamental (operation-type) machine groups of the system. At each stage one or more identical machines work in parallel. Each job has to be processed at each stage on any of the parallel concurrent machines.

Considering the production performance, both the allocation of machines and the sequence of jobs are of great importance. A great number of shop scheduling models are known in the literature, but most of them use only one performance measure. Usually the latest finishing time (make-span) of the released jobs appears as a goal function of optimization for Make to Stock (MTS) manufacturing. Frequently, one objective function related to due date plays the main role in scheduling models for supporting Make to Order (MTO) manufacturing. Only a

few of the models deal with multi-objective cases which are very important in flexible and agile manufacturing [2, 6, 8, 9].

The existing models in the operation research field often disregard the machine processing abilities, limited availability time frames of the machines, limited buffer capacities, and shared machine tools, that is why the improvement and extension of flexible shop models is justified. In order to consider the aforementioned important features of real scheduling problems we have focused on the simulation based scheduling approach. This paper presents our approach in which an execution-driven fast simulation is used to transform and reduce the problem spaces preserving important details.

3. Practice-oriented fine scheduling approach

The starting point of the new research was the Extended Flexible Flow Shop model (EFFS) developed by the first author of this paper [5]. The main goal of our new research is the aforementioned model which will be improved in order that the problem class EFFS can also represent the problems for smaller units. From the execution point of view, the units inherit the schedule of the job concerned but are moved along in the shop environment and have their own due date.

In this paper an integrated approach is proposed to solve this complex scheduling problem class as a whole without decomposition. In the approach, all the issues (batching, assigning, sequencing and timing) are answered simultaneously. For solving production scheduling problems in practice, a knowledge intensive searching algorithm has been developed based on execution-driven fast simulation, overloaded relational operators and multiple neighbouring operators. The core of the engine implemented explores iteratively the feasible solution space and creates neighbour candidate solutions by modifying the actual resource allocations, job sequences and other decision variables according to the problem space characteristics. The objective functions concerning candidate schedules are evaluated by producing a simulation which represents the real-world environment with capacity and technological constraints. In this execution-driven simulation, items, parts, units and jobs are passive and they are processed, moved, and stored by active system resources such as machines, material handling devices, manpower and buffers. The numerical tracking of the product units provides the time data of the manufacturing steps. The simulation process extends the pre-defined schedule to a fine schedule by calculating and assigning the time data. Consequently the simulation is able to transform the original searching space into a reduced space by solving the timing sub-problem. This is the part of the approach that encapsulates the dependency of real-world scheduling problems. Successful adaptation of the approach into practice is highly influenced by the efficiency of the simulation algorithm.

4. Application of the approach in practice

4.1. Motivation

The scheduling problem, which will be outlined in this section, is inspired by a real case study concerning only one of the plants of Electrolux-Lehel Ltd. specialized in refrigerator products (Jászberény, Hungary). The firm produces different types of refrigerators with variable series simultaneously. It is typical that the market centres or other distributors require very hard delivery due dates. Another important market trend is that the number of product variants is increasing. The companies are forced more and more to customize products to market demands and to important seasons. These requirements cause a tendency to decrease lot size, to develop better forecast, make more flexible production plans, and to use information technology, operation research and artificial intelligence methods for more efficient shop floor scheduling.

4.2. Problem description

The discrete manufacturing process examined produces various refrigerators as final products. The production planners create the actual production plan for the next time interval (typically one or two weeks) by using ERP system. External orders, forecasts, market trends, seasonal characteristics are considered in decision making. The planning phase is concerned with balancing supply and demand. The production plan generated defines the production program of the final products at the finishing technological step of the manufacturing processes. In other words, the production plan consists of internal production orders and each production order specifies the final product needed, the required quantity, the destination and the due date demanded. The destination means the finishing machine which is assigned to the production order for processing. According to the process plan of the product type, pre-defined technological steps must be executed on the component parts.

The shop contains different machine groups connected to each other in a given configuration (Figure 1). Each machine group contains a pre-defined number of machines. In essence, refrigerator manufacturing includes two fields that can be distinguished: the cabinet and door manufacturing processes. These sub-processes can be modelled and analysed separately. In the system, refrigerators are assembled on two finishing assembly lines (AL 1, AL 2). In this finishing step the cabinet and the door meet with many other component parts to become one final product. Both of them are very important from the viewpoint of overall stability. All the necessary components have to be available at a given time at the destination pre-defined. In order that this requirement should be satisfied, a detailed schedule for all the predecessors are created efficiently to synchronize all the manufacturing steps and sub-processes.

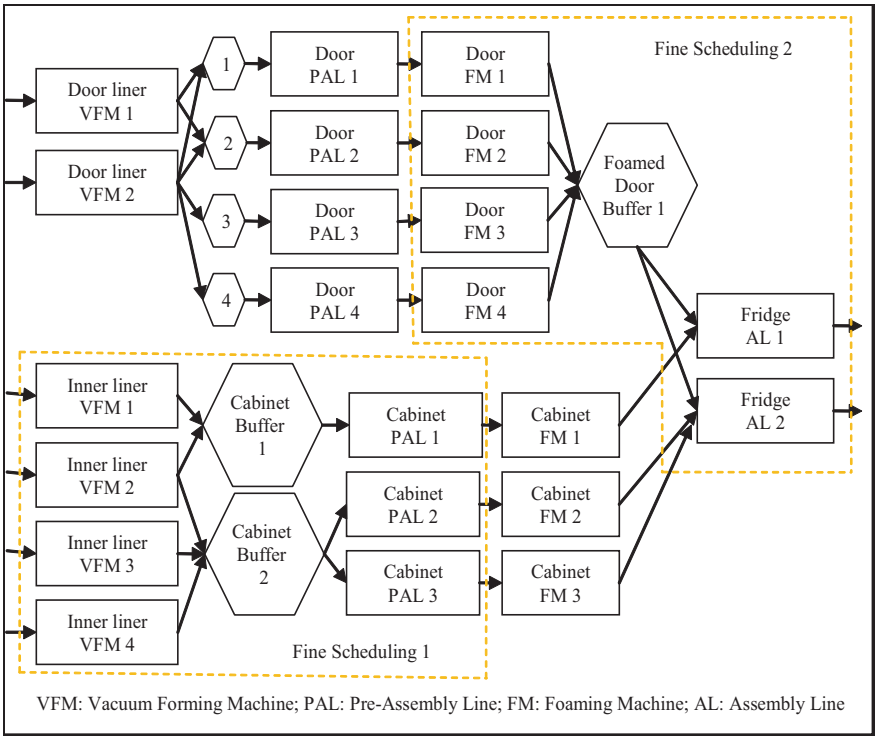


Figure 1. Scheme of refrigerator manufacturing

Based on the results of process analyses, it was clear that the scheduling of the inner liner vacuum forming and the door foaming play key roles. These sub-problems of the shop scheduling problem are denoted by *Fine Scheduling 1* and *Fine Scheduling 2* in Figure 1. The remaining part of this paper describes the applied model and solution method for detailed scheduling of door foaming.

The main purpose of the second R&D project was to develop a fine scheduler software in order to initiate detailed schedule for door foaming machines to realize the production plan and to synchronize the events on the finishing assembly lines of the manufacturing system.

4.3. Modelling the problem

4.3.1. Resource environment

In the problem labelled *Fine Scheduling 2* in Figure 1, z machines M_m $(m = 1, ...,z)$ have to process n jobs J_j $(j = 1, ..., n)$. A job J_j consists of b operations $O_{j1}, ...,O_{jb}$. The execution sequence is defined by the following precedence relations between the operations: $O_{jl} \rightarrow O_{j,l+1}$, for $l=1, ..., b-1$. Operation O_{jl} is not specified by a pre-defined processing requirement p_{jl}, because processing time depends on the

assigned resources. Each operation O_{jl} is associated with a set of machines $\mu_{jl} \subseteq \{M_1, ..., M_z\}$. O_{jl} may be processed on any of the machines in μ_{jl}. Usually, μ_{jl} is not one element set and μ_{jl} is not equal to the set of all machines.

Focusing on the door foaming scheduling, we have two operations: the first one is door foaming (FM) and the second one is final assembly (AL). The machine is dedicated to processing the AL of jobs because the input production plan determines the destination. For processing the door FM, the suitable machines are parallel. The foaming process of different types of doors can be executed on different foaming machines with various production intensities by using suitable tools. The pre-defined assignments and combinations mean hard constraints. In general, the number of available tools and machines is more than one for each door type. So the tools and machines belong to independent resource groups and can be used as shared resources. The foaming machine and the tools which are assigned to a given foaming operation O_{j1}, have to be available simultaneously for O_{j1} during the whole processing period. Scheduling problems of this type are called multi-processor task scheduling problems. A schedule is feasible if no two time intervals overlap on the same machine or the same tool, and if it meets a number of additional problem-specific characteristics.

The machines can work only within the time frames pre-defined. These machine availability intervals (shifts) have constant lengths (i.e. eight hours). Each machine has its own shift distribution. In order to minimize the cost, it is very important for shop floor control management that all the planned and started shifts are utilized as much as possible.

The capacity of the buffer shared among foaming machines and final assembly lines is strongly limited. As in the system different products are manufactured, we have to take into consideration the buffer capacity, which depends on the product types, because the volume of the buffer is constant but the sizes of the products are different. Consequently, in the calculation we use the relative utilization of the buffer.

4.3.2. Job characteristics

In order to create a final product, given components are taken through pre-defined processes. A job includes work-pieces and their operations. A job as a series of work-pieces must be scheduled in the manufacturing system. The actual work-piece depends on the operation to be performed. The operation is an elementary process of the manufacturing and changes one or more significant characteristics of the work-piece. A given sequence of operations must be executed on work-pieces to create final products. A sequence of operations means a technological step which can be performed on a single machine; similarly, a sequence of technological steps fulfilled by an integrated production or assembly line is an execution step.

Pre-emption of operations, technological steps or execution steps on a work-piece is not allowed in a classical sense, but a job execution on a machine can be suspended if the output buffer of the machine is full or the machine is unavailable for processing. Precedence relations between complete jobs are not specified, but restricted precedence chains of jobs (last operations) on finishing machines are given by the input production plan.

In the scheduling problem considered the set I of all jobs is partitioned into disjoint sets $I_1, ..., I_w$, where $I = I_1 \cup I_2 \cup ... \cup I_w$ and $I_f \cap I_g = 0$ for $f, g \in \{1, ..., w\}, f \neq g$. For any two jobs $J_x \in I_f$ and $J_y \in I_g$ to be processed on the same machine M_m, job J_y cannot be started before $sett_{mfg}$ time units after the finishing time of job J_x. Similarly, job J_x cannot be started before $sett_{mgf}$ time units after the finishing time of job J_y. The groups correspond to different types of products and $sett_{mfg}$ may be interpreted as a machine dependent changeover (or set-up) time. During the changeover period, the machine cannot process another job. If $f = g$, then $sett_{mfg} = 0$ for all $f, g \in \{1, ... , w\}$, $m \in \{1, ...,z\}$, and most of cases, if $f \neq g$, then $sett_{mfg} \neq sett_{mgf}$.

Generating a schedule for the following time horizon, it has to be considered that the machines can be loaded with unfinished tasks. So the input data set has to include the actual state variables of the system. It means that the effect of the last confirmed schedule must be considered when creating a new schedule.

4.3.3. Objective functions

In order to express the shop floor management's goals as criteria of a multi-objective optimization problem, we use seven objective functions to be minimalized. These are as follows:
- the number of tardy jobs,
- the sum of tardiness,
- the maximum tardiness,
- the number of set-up activities,
- the sum of set-up times,
- the average waiting rate of machines, and
- the average flow time of jobs.

4.4. Solving the problem

4.4.1. Decision variables

In the approach applied the job plays the role of the basic scheduling item. In order to create the fine schedule of the foaming machines, it is necessary for each job:
- to be assigned it to one of the suitable foaming machines,
- to fix its execution position in the queue for the assigned machine,
- to be assigned it to one of the suitable tools for use on the assigned machine, and

- to pre-set its starting time on the assigned machine.

To make decisions on these issues is very complicated. We developed a new approach in order to answer these questions. The main idea of this approach is a problem space transformation based on simulation. We use the following sets of independent decision variables to represent a candidate schedule as a solution:
- the sequence of job-machine assignments on each foaming machine,
- the specification of the availability time frames (shifts) of each foaming machine.

The availability time intervals of a given machine M_m are expressed by a pre-defined calendar (CAL_m). CAL_m is a list which stores the availability time frames specified by means of start time and end time values. The calendar elements of the foaming machines are decision variables, where the availability time frames of the final assembly machines belong to the set of constraints.

The decision variables of this reduced problem space form a simple schedule which will be extended to a fine schedule (detailed production program) by using simulation. The simulation, which is a fast execution-driven simulation of the simple schedule, answers the remaining issues concerning the tools needed and the starting time data of the execution steps. Consequently, the simple schedule determines the fine schedule. Sizes of production batches are formed dynamically by scheduling the jobs and executing the production units on machines.

To accelerate the simulation we use indexed data structures as attributes of the model objects. This memory model is based on indexes which are non-negative integer values assigned to the entities, to point to the position in the target object. The data model developed supports the association of two or more different type objects (i.e. machines and jobs). Before scheduling a builder method creates the full indexed data model which includes the valid technological and resource constraints and possible alternatives (i.e. job dependent sets μ_{jl} of machines).

4.4.2. Internal due dates

From the execution point of view it is allowed that the job consists of smaller production units (PU). These PUs inherit the schedule of the job concerned but are moved along in the shop environment. The internal due date of a given job on a given foaming machine is equal to the planned starting time of the same job on the pre-defined final assembly line. Considering that the machines can only work in accordance with pre-defined calendars, the internal due date of a given job refers to the first unit of the job in the strict sense. The exact due date of the other units of the job can be calculated by simulating the execution of the job on the final assembly line assuming that all of the units arrive in time. Using this procedure, the precise due date can be adjusted and assigned to each unit. These values are used as indirect input data of the scheduling problem. In the simulation and the evaluation of a candidate schedule, a given job will be delayed if either of its units is delayed.

4.4.3. Execution-driven simulation

An execution-driven simulation can be realized with a rule-based numerical simulation of the production to calculate the time data of the execution steps. Input data determine the production order, the jobs, the production units, the resources and the schedule to be executed. The schedule specifies the assignment of jobs and machines, and in addition defines the execution sequences of jobs on machines and the shift arrangement of machines.

Every job is represented by an individual model object (J_j) in the simulation. A J_j means a set of work-pieces of the same type. All of the work-pieces of a given J_j are processed on the same technological route by the same machines using the same tools. The route and the machines are chosen by the scheduler and defined in the schedule to be executed. A J_j consists of units U_i ($i=1, \ldots, v_j$). Every U_i represents one or more work-pieces (item series) to be moved as an individual atomic unit among machines. An execution step of a given unit on a machine means a processing task.

The main steps of the simulation are as follows:
- build and initialize the model objects,
- choose the next execution step (task) to be performed,
- simulate the execution of the active unit on the active machine.

The most important objects of the simulation model are the production orders, the jobs, the tasks, the machines, the buffer, the tools, the input schedule, the output fine schedule and the object of performance indicators. At the beginning of the calculation these objects are initialized with the actual values of the system state variables.

The execution-driven method calculates and stores the time data of the execution steps. On each machine the execution sequence of the assigned jobs is pre-defined. The main issue is how the limited resources (tools and buffer) affect the execution. To answer this issue, the simulation must perform all of the activities in a suitable sequence. This sequence cannot be pre-defined but it is part of the simulation. In an intermediate situation, the next execution step must be chosen from the set of candidate units. Each machine has a loading list and a pointer that shows the next unit to be processed according to the schedule. The pair of a given machine and its mentioned unit means a candidate execution step for processing if all the starting requirements are satisfied. These are as follows:
- the machine is not blocked by the buffer,
- the machine has finished its previous unit,
- the unit execution has been completed successfully on its previous machine,
- one of the suitable tools is available for processing.

The method chooses the candidate execution step which can be started the earliest. The machine and the unit associated with the chosen execution step become active

entities. The method calculates the time data (start time ST_{mi}, set-up time $SetT_{mi}$, processing time $ProcT_{mti}$, and completion time CT_{mi}) of the active unit on the active machine. The processing time ($ProcT_{mti}$) is determined by the work-piece quantity (q_i) of the unit, the tool and the unit (product type) dependent production rate (pr_{mti}) of the machine. The start time ST_{mi} of a given unit U_i on an assigned machine M_m is determined by the following values:
- the end time of the interval while the machine is blocked by the buffer (BT_{mi}),
- the end time of the interval while the tool is unavailable or engaged (e_t),
- the earliest release time of the unit (r_i),
- the completion time of the unit on the previous machine (ct_{pi}),
- the moving time of the unit from the previous machine (mt_{ipm}),
- the completion time of the previous unit on the machine (ct_{mh}),
- the unit-sequence dependent set-up time on the machine ($sett_{mhi}$),
- the availability time frames of the machine (CAL_m).

Focusing on the simulation of the execution step of unit U_i on the assigned machine M_m, the simplified description of the calculation can be seen in Figure 2 assuming that the set-up activity can be started on the machine before the unit arrives (a_{mi}).

$a_{mi} = ct_{pi} + mt_{ipm};$
$SetT_{mi} = sett_{mhi};$
$ProcT_{mi} = q_i / pr_{mti};$
$ST_{mi} = max(a_{mi} – SetT_{mi}, ct_{mh}, r_i - SetT_{mi}, BT_{mi}, e_t);$
$CT_{mi} = ST_{mi} + SetT_{mi} + ProcT_{mi};$
$Load\_STET\_to\_CAL(ST_{mi}, CT_{mi}, M_m);$

Figure 2. A simplified calculation of the time data of a given execution step

The function *Load_STET_to_CAL* loads the timeframe required by unit U_i on machine M_m. This allocation method inserts the set length time window [ST_{mi}, CT_{mi}] into the first suitable time frame of machine M_m. While the full size of the required time window does not fit in the candidate time interval, the time window is moved right to the next candidate time interval. This version of the load function represents that the execution step of the unit is not pre-empted in time.

Simulation of an execution step covers the handler of the tools and the buffer involved. The work-piece(s) of a given unit must be taken out of the buffer prior to the start of the execution on a final assembly line. If the execution step is finished successfully on a foaming machine, the work-piece(s) of the unit are to be put in the buffer. If the buffer is full, then the work-piece(s) stays on the foaming machine, therefore the next unit cannot be started and the foaming machine will be blocked. If the stock level in the buffer decreases below a given value, the blocked machine or machines will be released.

One of the suitable tools is allocated by the first unit of the job before starting the execution step. If more than one suitable tool is available at the same time, the earliest released tool will be chosen. The exclusive reservation of the chosen tool will be cleared by the last unit of the job.

The most important output data of the execution-driven simulation of the production are coded in a data object *MSTET* which stores the evaluated time data of all units. The simulation extends the pre-defined input schedule to a fine schedule by calculating and assigning *MSTET* in a short time. The performance of the fine schedule can be measured by calculating objective functions based on the data of units, jobs, and machines. In this way the simulation is able to transform the original searching space of the scheduling problem into a reduced space.

4.4.4. Search algorithm

For solving the scheduling problem in an integrated form addressed above, we developed an advanced multi-operator and multi-objective search algorithm based on overloaded relational operators, multiple neighbouring operators and a special taboo list which stores schedules in coded form (MOMOTS). Our approach is based on the taboo searching meta-heuristics that was first suggested by Glover [3] and has been used frequently for different combinatorial optimization problems.

```
MOMOTS
{ s0 ← Generate an initial solution;
  s* ← s0;
  Taboo_List ← NULL;
  while ( Stop criterion is not satisfied )
  { while ( Extension criterion is satisfied )
    {  Nc ← Choose the actual neighbouring operator(priority_list);
       s ← Generate a neighbour solution( s0 , Nc);
       if ( Taboo_List does not include ( s ) )
         {  Insert new taboo into the first position of Taboo_List ( s );
            if ( Number of Taboos > Maximum number )
               Delete the taboo from the last position of Taboo_List;
            if ( This is the first solution of the extension ( s ) ) sk ← s;
            else if ( s < sk ) sk ← s;
         }
    }
    s0 ← sk;
    if ( sk < s* ) s* ← sk;
  }
  return s*;
}
```

Figure 3. Multi-Operator and Multi-Objective Taboo Search (MOMOTS)

Our search algorithm variant (Figure 3) iteratively moves from an actual schedule s_0 to a candidate schedule s in the neighbourhood of s_0 until the stop criterion is satisfied. To reach and examine the unexplored regions of the search space, the method modifies the neighbourhood structure of each schedule as the search progresses. To escape local optimum, the method contains the schedules that have been visited in the recent past (less than a given number of moves ago) in a taboo list. Schedules in the taboo list are excluded from the neighbourhood of the actual schedule. A certain number of neighbours of the current schedule are generated at random successively by using priority controlled neighbouring operators. The operator can only modify the first execution step (decision variables) of jobs in the schedule (solution) because the second one is pre-defined by the input data (constraints). The applied neighbouring operators are as follows:

- operator N_1 moves a randomly chosen job elsewhere,
- operator N_2 moves a randomly chosen tardy job elsewhere,
- operator N_3 chooses a machine randomly and modifies the sequence of jobs on the machine by using a random length permutation cycle,
- operator N_4 exchanges two adjacent jobs on a machine which is chosen randomly,
- operator N_5 chooses a tardy job and moves left one position in the sequence,
- operator N_6 allows a denied calendar element randomly chosen on a machine,
- operator N_7 denies an allowed calendar element randomly chosen on a machine.

The operators listed create new candidate schedules by modifying the values of the decision variables of the initial schedule. The objective functions concerning candidate schedules are evaluated by the execution-driven simulation. The overloaded relational operator $<$ is used to compare the generated schedules according to multiple objective functions described in Section *4.3.3*. These objective functions are given so that:

$$f_k : S \to \Re^+ \cup \{0\}, \forall k \in \{1, 2, ..., K\}. \tag{4.1}$$

Coefficients w_k *(k=1, ..., K)* as input parameters support that the user may calibrate the actual priority of each f_k independently. Each w_k is an integer value within a pre-defined close range [*0, 1, ..., W*] and expresses the importance of f_k.

Let s_x, $s_y \in S$ be two candidate solutions. Function F is defined by (4.2) to express the relative quality of s_y compared to s_x as a real number.

$$F : S^2 \to \Re, F(s_x, s_y) = \sum_{k=1}^{K} (w_k \cdot D(f_k(s_x), f_k(s_y))). . \tag{4.2}$$

Function D defined by (4.3) means the comparison of s_x and s_y according to f_k.

$$D : \Re^2 \to \Re,\ D(a,b) = \begin{cases} 0, \text{if } \max(a,b) = 0 \\ \dfrac{b-a}{\max(a,b)}, \text{otherwise.} \end{cases} \tag{4.3}$$

Using (4.2) the relational operators are overloaded by (4.4):

$$(s_y\ ?\ s_x) := (F(s_x, s_y)\ ?\ 0). \tag{4.4}$$

Any of the relational operators (i.e. in C++ programming language: <, >, <=, >=, ==, !=) can be used between two solutions to compare them as two real numbers. For example: s_y is a better solution than s_x ($s_y < s_x$ is true) if $F(s_x, s_y)$ is less than zero.

After comparing candidate solutions in an actual loop, the best schedule becomes the initial solution of the next loop. When the scheduling process is finished or stopped by the user, the currently best known schedule is returned, so the method can be used in any-time working model.

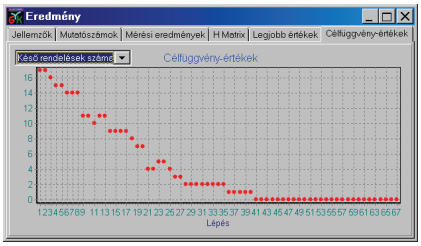
4.5. Some numerical results

For testing the proposed multi-objective predictive scheduling method we have used the data sets developed in Electrolux-Lehel Ltd. (Refrigerator Manufacturing Plant, Jászberény, Hungary), which represent real industrial problems.

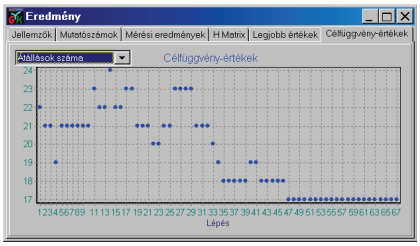
One of the case studies is summarized in Table 1. The main characteristics of the problem are as follows: scheduling time horizon is one week, time unit is one minute, number of jobs is 30, and number of production units is 16201.

Table 1. Sample Results of Multi-Objective Scheduling

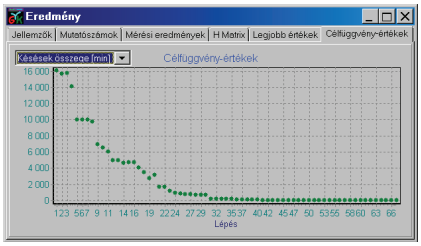
Objective function priority (w_k)	**Objective function (f_k)**	**Initial solution (s_0)**	**Best solution (s^*)**
5	f_1: number of tardy jobs	18	0
5	f_2: sum of tardiness [min]	16044.20	0
10	f_3: maximum tardiness [min]	2289.68	0
5	f_4: number of set-up activities	23	17
5	f_5: sum of set-up times [min]	760	500
5	f_6: average utilization rate of the machines [%]	38.12	91.55
5	f_7: average flow time of jobs [min]	1851.28	1419.48



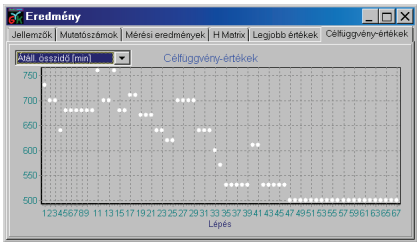
f_1: number of tardy jobs



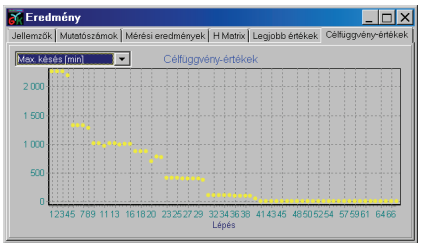
f_4: number of set-up activities



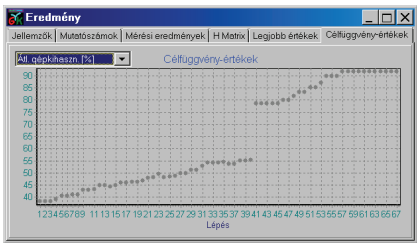
f_2: sum of tardiness [min]



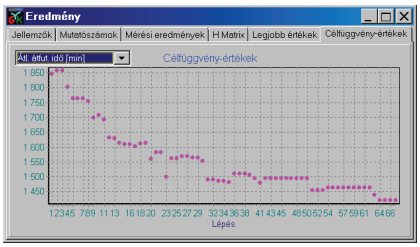
f_5: sum of set-up times [min]



f_3: maximum tardiness [min]



f_6: average machine utilization rate [%]



f_7: average flow time of jobs [min]

Figure 4. Actual values of the objective functions represented in the searching steps

Parameters of the searching algorithm are as follows: the maximum number of elements in taboo list is 150, the number of neighbour solutions in extension is 50, and the priority of each neighbouring operator is 1. In this case the computational time of the solution process is approximately 50 sec. The software was coded in C++ language. Concerning the running test environment we have; Intel^{R} Core^{TM} 2 Duo T9550 2.66GHz CPU, Windows Vista OS, 4GB RAM. The actual values of

the objective functions represented in the searching steps can be seen in Figure 4. All applications of the objective functions occur simultaneously. The diagrams (screen shots) are generated by our software (with a Hungarian user interface).

The proposed method can solve the examined problems effectively in a short time. The software developed is used in daily practice at shop floor control level of the plant.

5. Conclusions

The paper describes the proposition and application of a practice-oriented approach for solving multi-objective scheduling problems. It is based on execution-driven simulation and use of relational operators for comparing qualities of schedules in search algorithms. After developing the software, the concept is successfully tested on extended flexible shop problems considering multiple objectives and constraints originating from an industrial environment. Scheduling based on simulation can consider exactly what the actual manufacturing system should perform in the planned time horizon. In this approach, each schedule created for the shop is a feasible solution, because all of the hard constraints are considered. The results obtained and the independent nature of the approach encourage the application of the method in other multi-objective optimization problems.

Acknowledgements

This research was carried out as part of the TAMOP-4.2.1.B-10/2/KONV-2010-0001 project with support by the European Union, co-financed by the European Social Fund.

REFERENCES

[1] BAKER, K.: *Introduction to Sequencing and Scheduling.* 1st ed. Canada: John Wiley & Sons, 1974.

[2] BAYKASOĞLU, A., ÖZBAKIR, L., DERELI, T.: *Multiple Dispatching Rule Based Heuristic for Multi-Objective Scheduling of Job Shops Using Tabu Search*. In Proceedings of the 5th International Conference on Managing Innovations in Manufacturing, pp. 396-402, Milwaukee, USA, 2002.

[3] GLOVER, F.: *Tabu Search: a Tutorial.* Interfaces, 20, 74-94, 1990.

[4] KULCSÁR, GY., ERDÉLYI, F.: *A New Approach to Solve Multi-Objective Scheduling and Rescheduling Tasks*. International Journal of Computational Intelligence Research, 3, (4), pp. 343-351, 2007.

[5] KULCSÁR, GY., KULCSÁRNÉ, F. M.: *Solving Multi-Objective Production Scheduling Problems Using a New Approach.* Production Systems and Information Engineering, A Publication of the University of Miskolc, 5, 81-94, 2009.

[6] LOUKIL, T., TEGHEM, J., TUYTTENS, D.: *Solving Multi-Objective Production Scheduling Problems Using Metaheuristics*. European Journal of Operational Research, 161, pp. 42-61, 2005.

[7] QUADT, D., KUHN, H.: *A Taxonomy of Flexible Flow Line Scheduling Procedures*, European Journal of Operational Research, 178, pp. 686-698, 2007.

[8] SBALZARINI, L. F., MÜLLER, S., KOUMOUTSKOS, P.: *Multiobjective Optimization Using Evolutionary Algorithms*. In Center of Turbulence Research, Proceedings of the Summer Program 2000, pp. 63-74, 2000.

[9] SMITH, K. L., EVERSON, R. M., FIELDSEND, J. E.: *Dominance Measures for Multi-Objective Simulated Annealing*. In Proceedings of Congress on Evolutionary Computation, pp. 23-30, 2004.

[10] WANG, W.: *Flexible Flow Shop Scheduling: Optimum, Heuristics, and Artificial Intelligence Solutions*, Expert Systems, 22, (2), pp. 78–85, 2005.