A New Approach to Solve Multi-Objective Scheduling and Rescheduling Tasks

Gyula Kulcsár[1] and Ferenc Erdélyi[2]

[1]University of Miskolc, Department of Information Engineering,
Egyetem Road, Miskolc-Egyetemváros 3515, Hungary
kulcsar@ait.iit.uni-miskolc.hu

[2]University of Miskolc, Department of Information Engineering,
Egyetem Road, Miskolc-Egyetemváros 3515, Hungary
erdelyi@ait.iit.uni-miskolc.hu

***Abstract*: In this paper a new approach is presented to solve multi-objective production scheduling and rescheduling problems. Furthermore the relationship between production goals and heuristic solving methods in an extended flexible flow shop environment is also investigated. We define production goals by specifying objective functions and apply special production constraints for the extended flow shop scheduling model. We focus on creating near-optimal feasible schedule considering multiple objectives. The proposed method is based on the well-known tabu search meta-heuristic. Moreover we use an advanced structure of the tabu list, new relational and neighbourhood operators for multi-objective aim. At the end we present some computational experiments on small and large size problem instances.**

***Keywords*:** scheduling, rescheduling, multi-objective optimization, tabu search, extended flexible flow shop**.**

I. Introduction

In a multi-objective scheduling problem class, we wish to find such a feasible schedule which optimizes a set of objective functions and subjects to a set of well-defined special constraints. The task is NP-hard therefore the "optimal" schedule is defined as a result of evolving process in which an engineer and/or a computer program may reach the desired (and compromised) values of the scheduling variables.

Meta-heuristics (i.e.: genetic algorithms, simulated annealing and tabu search) have become successful methods for optimization problems that are too complex to be solved using deterministic techniques (see i.e. [1], [8], [6], [12]). To solve a multi-objective scheduling problem, it is necessary to answer an additional important question: What does "good" schedule mean? It is not easy to specify the answer in mathematical form because in real-life situations there are many objectives (based on cost, delivery capability, machine utilization rate, stock level, etc.) and they are usually conflicting. The actual importance of objectives can vary frequently in time. A typical appearance of this problem in customized mass production is the decisions emerged at shop floor scheduling and rescheduling tasks.

This paper describes a new approach which can be used in well-known meta-heuristics for comparing schedules in accordance with multiple objectives. An application of the approach is presented to the multi-objective extended flow shop scheduling and rescheduling problems.

II. Related Works on Multi-Objective Optimization

Optimization problems often involve more then one aspect so it is required to use multiple criteria simultaneously. For these optimization problems the goal is:

$$\min_{s\in S}\left[f_1(s),\ldots f_k(s),\ldots f_K(s)\right],\ \forall k \in \{1,2,\ldots,K\} \qquad (1)$$

where s is a solution, S is the set of feasible solutions and functions f_k are the objectives. An objective is a measure to evaluate the quality of the solution from a given point of view.

In the literature, different approaches can be found considering multi-objective scheduling problems, as they are surveyed i.e. in [11] and [6]. Four main approaches are as follows:

- Simultaneous method (or Pareto approach) aims to generate the complete Pareto set or to approximate a set of efficient solutions. A set of solution is said to be Pareto set if passing from solution s_A to another solution s_B in the set, any improvement in one of the objective functions from its current value would cause at least one of the other objective functions to deteriorate from its current value.
- Weighting objectives method creates a weighted linear combination of the objectives to obtain a single function, which can be solved using any single optimization method.
- Hierarchical optimization method allows the decision maker to rank the objectives in a descending order of importance, from *1* to *K*. Each objective function is then minimized individually subject to a constraint that does not allow the minimum for the new function to exceed a prescribed fraction of a minimum of the previous function.
- Goal programming method takes the objectives into constraints which express satisfying goals. The aim is to find a solution which provides good values of pre-defined goals for each objective.

III. A New Method based on Relative Quality

According to our new idea the relative goodness of a solution is more important than the absolute goodness of one. The base of the method is the following: we measure the relative goodness of the selected solution by comparing it with another solution in the feasible region.

Let S be the search space under consideration. It is the set of all possible solutions to our problem. Suppose that we have a number of objective functions $f_1, \ldots, f_K$ such that:

$$f_k : S \to \Re^+, \ \forall k \in \{1,2,\ldots, K\} \quad (2)$$

The problem is to find an $s \in S$ that minimizes every $f_k(s)$. This is known as a multi-objective optimization problem. In many cases, it is not possible to find a solution to a multi-objective optimization problem. Successfully minimizing one of the component objective functions will typically increase the value of another one.

So we must find solutions that represent a compromise among the various criteria used to evaluate the quality of solutions.

More formally, let $s_x, s_y \in S$ be two solutions. We define the function F to express the quality of s_y compared to s_x as a real number:

$$F : S^2 \to \Re,$$
$$F(s_x, s_y) = \sum_{k=1}^{K} (w_k \cdot D(f_k(s_x), f_k(s_y))). \quad (3)$$

The definition of function D is the following:

$$D : \Re^2 \to \Re, \ a, b \in \Re,$$
$$D(a,b) = \begin{cases} 0, \text{if } \max(a,b) = 0 \\ \dfrac{b-a}{\max(a,b)} \cdot 100, \text{ otherwise} \end{cases}. \quad (4)$$

The *max(a,b)* used in (4) denotes an operator:

$$\max : \Re^2 \to \Re,$$
$$\max(a,b) = \begin{cases} a, \text{ if } a > b \\ b, \text{ otherwise} \end{cases}. \quad (5)$$

Moreover, to express the importance of any component objective function f_k, we use the coefficient w_k which is an integer value within range *[0, 1, ..., W]*. It is allowed that the decision maker sets the actual priority of each objective function independently.

Using the function F, two solutions $s_x, s_y \in S$ are compared as follows:

s_x is better solution than s_y ($s_x < s_y$ is true) if

$$F(s_x, s_y) > 0. \quad (6)$$

s_x and s_y are equal ($s_x = s_y$ is true) if

$$F(s_x, s_y) = 0. \quad (7)$$

s_x is worse solution than s_y ($s_x > s_y$ is true) if

$$F(s_x, s_y) < 0. \quad (8)$$

These definitions of the relational operators are suitable for applying in meta-heuristics like tabu search, simulated annealing and genetic algorithms to solve multi-objective combinatorial optimization problem.

IV. An Application to Extended Flexible Flow Shop Scheduling Problem

A. Shop Floor Scheduling in Customized Mass Production

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 in flexible 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: for satisfying their accepted orders and utilizing their manufacturing resources. In this "customized mass production" paradigm the firms plan their production partially for external direct orders arriving from logistic or shopping centres but to reach better delivery dates they must make forecast for manufacturing semi-finished products and buying materials with long external lead time.

In this paper, we are focusing on an extended flexible flow shop model with alternative routes, unrelated parallel machines and limited resource availability time intervals. The problem is inspired by a real case study concerning a multinational firm specialized in lighting products in Hungary. 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, detailed production schedule of the manufacturing and assembly processes.

In the manufacturing-assembly system different products may be produced. Each production order includes the identifier of the product, the required quantity and the demanded due date. At the shop floor level, product-pallets can be moved between automated machines (production-lines). Each pallet consists of a pre-decided number of products. Each production order is identified to be consisting of a particular number of pallets. In the system, the basic handling unit is the product-pallet. It is worth to schedule pallets, so one pallet means one job. Each job has four main attributes: 1.) the type of the final product, 2.) the quantity of the products to be processed, 3.) the start time (the earliest time when all of the required material available in the needed quantity) and 4.) the due date demanded. Each job has to visit a given number of technology steps in a specified sequence. (The types of manufacturing processes can be seen on Figure 1.) A technology step may include more sub-operations, but no pre-emption is allowed at the level of the technology steps. Typical technology steps may be preparing, manufacturing, quality checking and packaging.

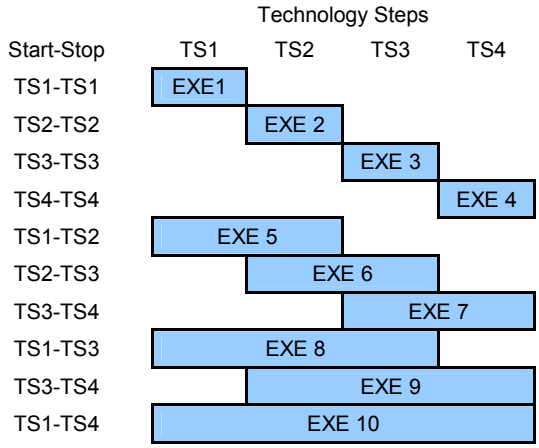

Figure 1. Types of Manufacturing Processes (EXE) in case of four Technology Steps

The workshop contains different machine groups connected to each others in a given configuration (Figure 2.). Each machine group contains a pre-defined number of machines. In a given machine group, each machine can process the same execution step which is a well-defined set and sequence of technology steps. The machines are not continually available for processing the pallets, therefore they have one or more non-availability time-intervals. In addition, each machine may have different production rates (quantities producible per time unit) for different products. Similarly, each machine may be characterized by product sequence dependent setup times (time delay to changeover from one product type to another product type).

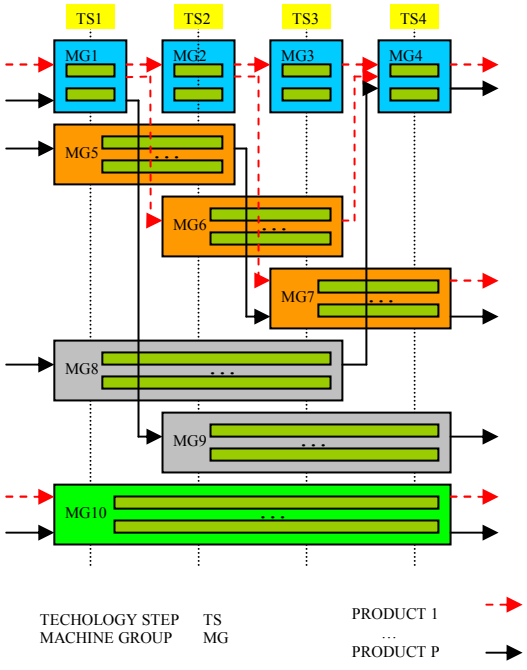

Figure 2. Extended Flexible Flow Shop Model

A given final product or a semi-finished product can be produced differently, because there are different execution routes on which the required components are taken through becoming the specified product. In detail, the sequence of execution steps is called execution route. So each execution route includes all the technology steps required in order that the 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.

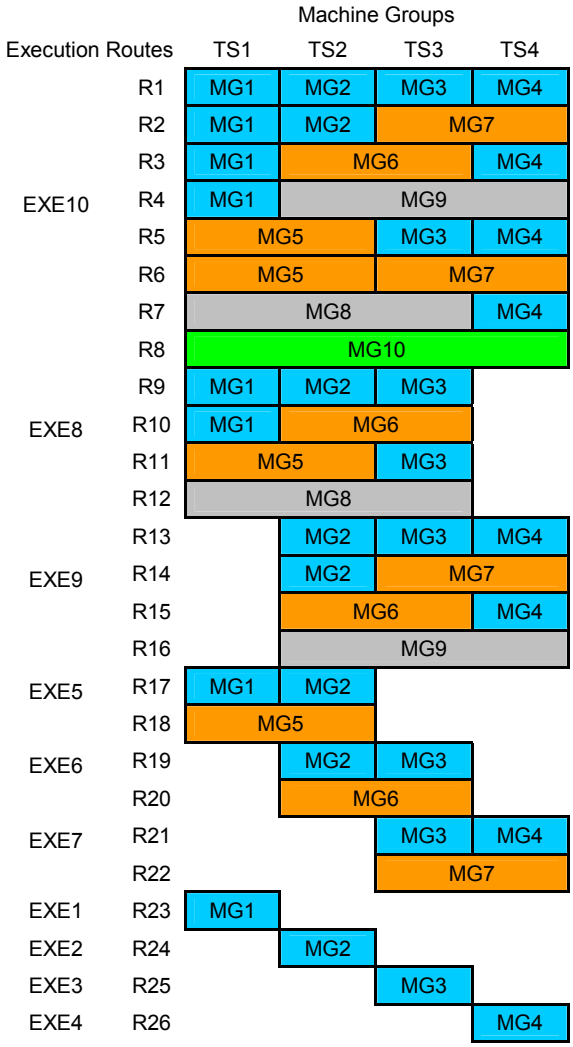

Figure 3. Execution Routes in case of four Technology Steps

At the start time of the scheduling process the machine lines has already been loaded but the actual state of the system is known. It means that the effect of the last confirmed schedule must be considered in the new schedule.

B. An Extended Flexible Flow Shop Model

Customized mass production requires advanced functions of Manufacturing Execution Systems (MES). Concentrating on the Shop Floor Scheduling functional module of the MES it comes into view that lots of flexible flow shop model exist in the literature (see i.e. [9], [13], [10], [2], [14]) but they do not consider all the requirements of customized mass production. The conventional flow shop model has to be extended to a new model that supports execution steps, alternative technological routes, and unrelated parallel machines with different capabilities. Sequence dependent setup times, special job characteristics and different objective functions are also considered at the same time. In order to formulate a new class of scheduling problem, the well-known formal specification $\alpha|\beta|\gamma$ is used, where:

- α denotes machine environment descriptors,
- β denotes processing characteristics and constraints,
- γ denotes the list of objective functions.

The EFFS scheduling model can be described as follows:

$$Fx, M_g, Q_{i,m}, Set_{i,j,m}, Cal_m \left| R_i, D_i, Exe_i, A_i \right| [f_1, ..., f_K]. \quad (9)$$

$\alpha = [\alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5]$

α_1: The type of operation (technology step) sequence at shop floor level. *F*: Flow Shop, *x*: the maximum number of operations.

α_2: The type of machine environment. M_g: group of multi-purpose machines which can execute one or more operations in a given sequence. Each machine group can include distinct parallel machines.

α_3: The type of alternative machines. $Q_{i,m}$: unrelated parallel machines with job dependent production rates.

α_4: The type of machine setups. $Set_{i,j,m}$: job sequence and machine dependent setup times.

α_5: Special resource constraint. Cal_m: machine availability time intervals.

$\beta = [\beta_1, \beta_2, \beta_3, \beta_4]$

β_1: Constrained released time of the jobs. R_i: the earliest start time of the jobs.

β_2: Constrained due date of the jobs. D_i: the latest completion time of the jobs.

β_3: The type of manufacturing processes. Exe_i: required type and sequence of technology steps for jobs.

β_4: Constrained resource assignments. A_i: set of suitable machines for the jobs.

$\gamma = [\gamma_1, \gamma_2, ..., \gamma_K]$

A scheduling objective is a measure to evaluate the quality of a certain schedule. In real-life situations, there are many different objectives.

For due date related objectives, we assume that there are production orders which consist of jobs J_i ($i = 1, 2, ..., N$) and manufacturing tasks Q_t ($t = 1, 2, ..., Z$). Each job i has due date D_i. The completion time of job i is denoted by C_i. The lateness of the job i is:

$$L_i = C_i - D_i, \quad (10)$$

and the tardiness of the job i is:

$$T_i = \max(0, L_i). \quad (11)$$

In current study, five objective functions are considered for demonstrating multi-objective predictive scheduling. These are as follows: the number of tardy jobs (12), the sum of tardiness (13), the maximum tardiness (14), the number of setups (15), the maximum completion time (16).

$$f_1 = |\{J_i \mid T_i > 0\}|. \quad (12)$$

$$f_2 = \sum_i T_i. \quad (13)$$

$$f_3 = \max_i (T_i). \quad (14)$$

$$f_4 = |\{O_t \mid setuptime_t > 0\}|. \quad (15)$$

$$f_5 = \max_i (C_i). \quad (16)$$

C. *Heuristic Method for Creating Near-Optimal Schedule*

The extended flexible flow shop scheduling problem (EFFS) is difficult to solve because of its combinatorial nature. The model inherits the difficulties of the classical flow shop (FS) and the flexible flow shop (FFS) models. Additionally, numerous strange features appear because of special extensions. The scheduling task consists of batching, assigning, sequencing and timing parts. We developed a new integrated approach to solve all the sub-problems as a whole without decompositions. In our approach all the issues are answered simultaneously (Figure 4.).

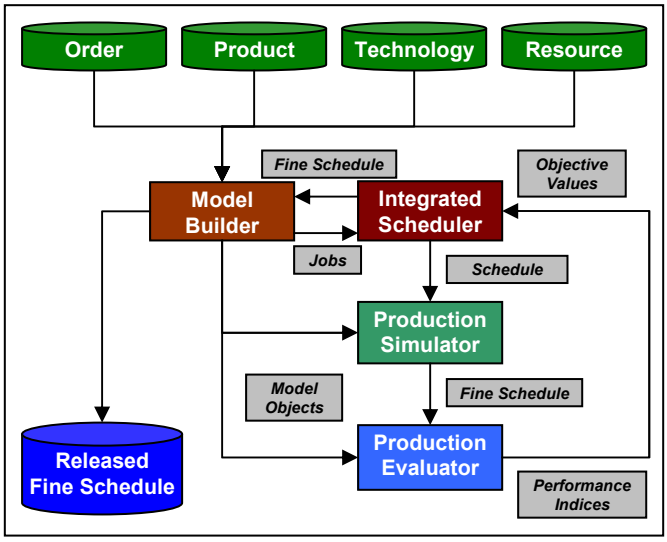

Figure 4. Integrated Production Scheduler

For solving the scheduling problem in an integrated form addressed above, we use an advanced tabu search algorithm based on simultaneous production simulation, overloaded relational operators, multiple neighbouring operators and special tabu list which stores schedules in coded form.

To accelerate the computation we use indexed arrays in our model. In these arrays, there are no full length identifiers and attributes of entities (i.e.: orders, 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. In any of indexes of a given array, we can use any value of the same array or another array. The developed data structure supports the association of two or more different type arrays. The model builder creates the full indexed data model which includes the possible technology and resource alternatives.

In our model the product-pallet plays the role of the basic scheduling unit. Each production order is divided into pallets that mean individual jobs. The production batch sizes are formed dynamically by scheduling the jobs on machines.

In order to create a detailed schedule for the production of each order, it is necessary for each job:

1. to assign to one of the possible routes,
2. to assign to one of the possible machines at each possible machine group according to selected route,
3. to fix its position in the queue of each selected machine,
4. to fix its starting time on each selected machine.

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

- Constructive algorithms.
- Iterative improvement algorithms.

This paper outlines one of these scheduling algorithms based on tabu search technique (TS). TS principle was first suggested by Glover [3]. It is a heuristic problem independent optimization method.

Our tabu search variant is a special search procedure which iteratively moves from a schedule s_x to a schedule s_y in the neighbourhood of s_x, until some stopping criterion has been satisfied. To explore regions of the search space that would be left unexplored by a simple local search procedure and escape local optimality, tabu search modifies the neighbourhood structure of each schedule as the search progresses. Our tabu list contains the schedules that have been visited in the recent past (less than a given number of moves ago). Schedules in the tabu list are excluded from the neighbourhood of the actual solution.

```
Multi-Objective Tabu Search
{
s0 = Generate an initial solution ();
s* = s0;
Tabu List = NULL;
while ( Stop criterion is not satisfied () )
{
 while ( Extension criterion is satisfied () )
 {
  s = Generate a neighbour solution (s0);
  if ( Tabulist do not include ( s ) )
     {
      Insert tabu into the first position of Tabu List ( s );
      if ( Number of Tabus () > Maximum number () )
        Delete tabu from the last position of Tabu List( );
      if ( This is the first solution of the extension ( s ) )
        s_k = s;
      else
           if ( F( sk, s) < 0 )
             sk = s;
      }
  }
  s0 = sk;
  if ( F( s*, sk ) < 0 )
    s* = sk;
 }
 return s*;
}
```

A certain number of neighbours of the current schedule are generated random successively by using priority controlled neighbouring operators. The applied neighbouring operators are as follows:

- N1 operator removes an order randomly chosen from the schedule then inserts all the jobs of the order as a whole.
- N2 operator removes a late order randomly chosen from the schedule then inserts all the jobs of the order as a whole.
- N3 operator modifies the sequence of jobs on a randomly chosen machine by using random length permutation-cycle.
- N4 operator removes a late job randomly chosen from the schedule then redefined the manufacturing tasks of the job by assigning resources and finally inserts the job into random position on each related machine.
- N5 operator works similarly to the operator N4 but the target job is chosen from all jobs.

These operators create new feasible schedules by modifying resource allocations and job sequences. The objective functions concerning schedules are evaluated by a production simulator which represents the machines with their capacity and the technological constraints. The simulation means rule based numerical tracking of the production to calculate the time data of the manufacturing steps (starting time, setup time, processing time and finishing time). The simulator extends the predefined schedule (job-resource assignments and job sequences on machines) to a fine schedule by calculating and assigning time data. The simulation is able to transform the original searching space to a reduced space.

Production evaluator measures the performance of the fine schedules by calculating management indices based on job, order and machine data. The function $F(s_x,s_y)$ proposed in section III is used to compare the generated schedules according to multiple objectives described in section IV.B. 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.

V. Novel Model for Solving Rescheduling Task

In practice, generating high quality predictive schedule according to multi-objectives is not enough because many unexpected events require the revision and modification of the released schedule. In the following, we shortly deal with some practical issues of the rescheduling task.

Rescheduling is a process of updating an existing production schedule in response to disruptions or creating a new one if the current schedule has become infeasible. Different type of uncertainty can occur i.e. machine failure or breakdown, missing material or components, under estimation of processing time, job priority or due date changes and so on (see in detail i.e. [5]). Different rescheduling methods can be used according to the effects of the unexpected events: time shift rescheduling, partial rescheduling or complete rescheduling. Time shift rescheduling postpones executions of certain tasks and jobs in time, but their resource assignments and sequences are not changed. Partial rescheduling modifies only jobs and resources affected by the disruption. Complete rescheduling generates a new feasible schedule. These methods are presented in detail in [4].

In order to solve the rescheduling task we use multi-objective searching algorithms similarly to the predictive scheduling described in section IV.C. The aim of rescheduling is to find a schedule which 1.) considers the modified circumstances, 2.) is near-optimal according to some predefined criterion and 3.) is as close as possible to the original one.

Rescheduling methods are required to consider new demands added to predictive scheduling problem. The last released schedule appears as a new input element of the rescheduling system and it is very important to preserve this initial schedule as much as possible to maintain the system stability. For this purpose, we defined qualitative indices (i.e. related

to setup and due date) for supporting comparison of schedule changes. For examples, we use the following indices:

- Number of the jobs having modified execution route.
- Number of the jobs having at least one modified parallel machine.
- Number of jobs having late status deviation.
- Number of the new late jobs.
- Number of the orders which have at least one job with modified execution route.
- Number of the orders which have at least one job with at least one modified parallel machine.
- Number of orders having late status deviation.
- Number of the new late orders.
- Number of machines with setup deviation.
- Number of machines with new setup.

Such performance indices can be considered as objective functions of rescheduling and they can be used by multi-objective scheduling methods proposed in the paper.

Lots of special rescheduling constraints have to be considered. For example:

- All jobs which are already finished when the rescheduling process starts are not changeable but can affect the other jobs and orders.
- The manufacturing tasks of jobs running at the starting time of the rescheduling process must not be interrupted and their possible execution route and parallel machines (and other alternatives) can differ from the original possibilities.
- Finished and running jobs on resources are known therefore they have to be regarded in the future.
- All production orders starting after the rescheduling process can be considered in their original status.

To satisfy such constraints we use freezing techniques. The main classes of these techniques are as follows: 1.) to freeze jobs, 2.) to freeze production orders and 3.) to freeze machines.

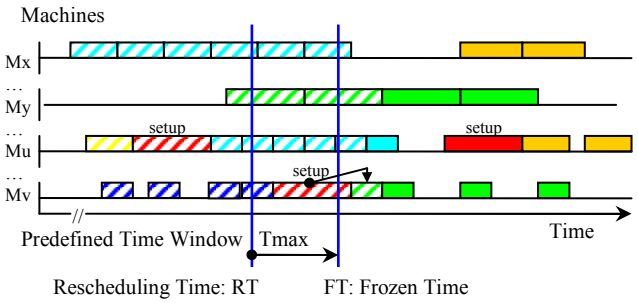

Figure 5. Machine Oriented Gantt Chart

The rescheduling problem can be solved by applying multi-objective searching algorithm with the proposed extensions.

To increase the flexibility and effectiveness of the rescheduling process, we developed a software tool for supporting the user interaction. Graphical user interface provides useful charts, diagrams, tables to show aggregate and detailed information of the production fine schedule. Rescheduling process is also scrutinized and checked on the screen, the user can modified at runtime the control priority and parameters of searching algorithms. In addition, the user can suspend the automatic process and edit the actual schedule by using available operation tools manually. The outlined mixed rescheduling approach based on human and artificial intelligence can yield significant performance improvements.

VI. Computational Results

For testing the proposed multi-objective predictive scheduling method we used sample data sets created by problem generator. The generator produces random problem instances with sizes and characteristics specified by the user. The main parameters are summarized in Table 1. Uniform distribution specified by average value and maximum deviation (in percent) was applied all cases. We used EFFS instances with four technology steps.

The applied objective functions were the number of late orders (LOrder), the number of late jobs (LJob), the sum of tardiness (sumT), the maximum tardiness (maxT), the number of setups (Set) and the maximum completion time (maxC).

Parameters
Number of setup types
Setup time
Number of parallel machines in a machine group
Machine speed
Number of calendar elements
Maximum start time of the first calendar element
Length of machine availability intervals
Length of machine unavailability intervals
Number of production orders with EXE type
Number of the jobs of a production order
Number of the products of a job
Value of the constraint start times
Length of the constraint time windows
Special machine requirements of the production orders

Table 1. Parameters of the Problem Generator

We tested the algorithm on small size instances and large size instances, as well. Some typical results are summarized in Table 3. – Table 5.

The actual values of the objective functions represented in the searching steps can be seen on Figure 6. – Figure 10.

Based on the results of the tests which are executed on small size problem instances, we can say that the proposed method is able to find optimal solution. The results are validated by comparing with the result of an optimal method based on enumeration technique. The proposed method can be able to solve the large size problems effectively in a reasonable amount of time. Demo version of the implemented software with built-in problem generator and the tested input data are available: http://ait.iit.uni-miskolc.hu/~kulcsar/ijcir_sch.htm.

Personal Computer	
OS	Microsoft Windows XP
CPU	Intel Pentium 4 2.8 GHz
RAM	512MB DDR333

Table 2. Test Environment

Average Performance Data (10 iterations)						
Input_file	si1.txt					
Prob_Sizes	Machines: 11 Orders: 16 Jobs: 18 Setup_Types: 3					
Search_ Pars	maxSTEP: 200 maxLOOP: 100 minLOOP: 10 maxImpless: 20 maxTABU: 150					
Nx_op_pri	N1: 1 N2: 1 N3: 1 N4: 1 N5: 1					
Obj_F	LOrder	LJob	sumT	maxT	Set	maxC
Obj_Pri	7	8	10	10	7	5
Rand	4	5	627	175	15	523
Search	***0***	***0***	***0***	***0***	***13***	***430***
Enum	0	0	0	0	13	430
Num_of_Feasible_Sch	10501632					
Num_of_Optimal_Sch	144					
Input_file	Si2.txt					
Prob_Sizes	Machines: 11 Orders: 14 Jobs: 17 Setup_Types: 3					
Search_ Pars	maxSTEP: 200 maxLOOP: 100 minLOOP: 10 maxImpless: 20 maxTABU: 150					
Nx_op_pri	N1: 1 N2: 1 N3: 1 N4: 1 N5: 1					
Obj_F	LOrder	LJob	sumT	maxT	Set	maxC
Obj_Pri	7	8	10	10	7	5
Rand	3.5	5.5	488.5	175	13.5	485.5
Search	***2***	***4***	***182***	***88***	***12***	***430***
Enum	2	4	182	88	12	430
Num_of_Feasible_Sch	22334976					
Num_of_Optimal_Sch	48					

Table 3. Results on Small Size Problems

Average Performance Data (10 iterations)						
Input_file	li1.txt					
Prob_Sizes	Machines: 119 Orders: 393 Jobs: 2173 Setup_Types: 4					
Search_ Pars	maxSTEP: 2000 maxLOOP: 100 minLOOP: 10 maxImpless: 40 maxTABU: 150					
Nx_op_pri	N1: 1 N2: 1 N3: 1 N4: 1 N5: 1					
Obj_F	LOrder	LJob	sumT	maxT	Set	maxC
Obj_Pri	7	8	10	10	7	5
Rand	277.5	992.1	1.3e+06	4241.1	1779.6	4685.2
Search	***2.1***	***3.2***	***48***	***21.3***	***248.7***	***1034.7***
Elapsed_Time	2 min 55 s					
Input_file	li2.txt					
Prob_Sizes	Machine: 56 Orders: 424 Jobs: 2360 Setup_Types: 4					
Search_ Pars.	maxSTEP:2500 maxLOOP:100 minLOOP:10 maxImpless:40 maxTABU:150					
Nx_op_pri	N1: 1 N2: 1 N3: 1 N4: 1 N5: 1					
Obj_F	LOrder	LJob	sumT	maxT	Set	maxC
Obj_Pri	7	8	10	10	7	5
Rand	338.4	1382.6	3.7e+06	8360	1543.1	8870.4
Search	***15.6***	***60.3***	***6110.9***	***300.5***	***306.4***	***1709.7***
Elapsed_Time	3 min 49 s					

Table 4. Results on Large Size Problems

Quality of the Best Solution Found in the Iterations						
Input_file	li1.txt					
Obj_F	LOrder	LJob	sumT	maxT	Set	maxC
Obj_Pri	7	8	10	10	7	5
S_1	2	3	33	14	243	998
S_2	2	3	46	22	234	1068
S_3	2	3	46	22	250	1035
S_4	2	3	46	22	265	1032
S_5	2	3	33	14	228	1031
S_6	2	3	46	22	228	995
S_7	2	3	46	22	237	1055
S_8	2	3	46	22	244	1058
S_9	3	5	92	31	304	1014
S_10	2	3	46	22	254	1061

Table 5. Distribution of the Best Solutions (li1 test)

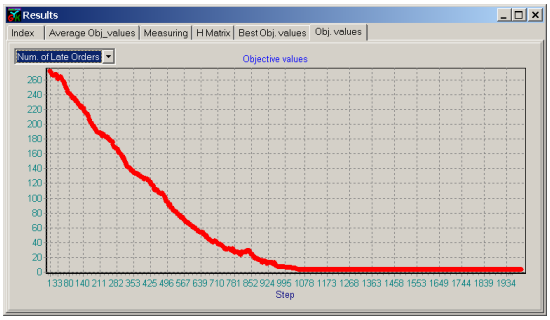

Figure 6. Number of Late Orders (LOrder in li1 test)

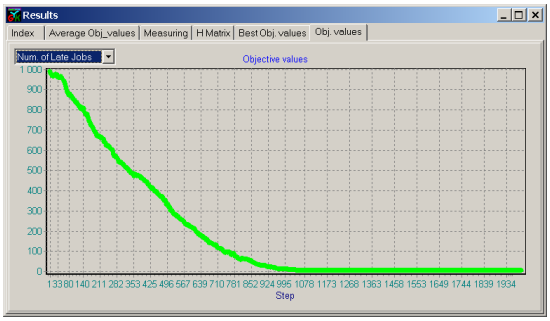

Figure 7. Number of Late Jobs (LJob in li1 test)

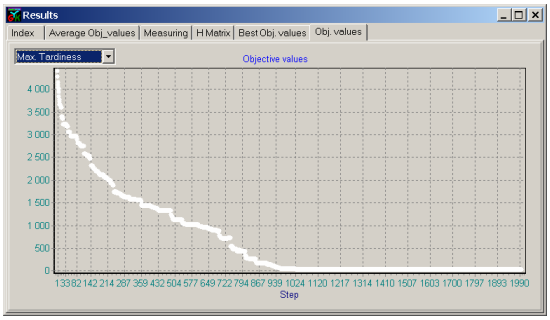

Figure 8. Maximum Tardiness (maxT in li1 test)

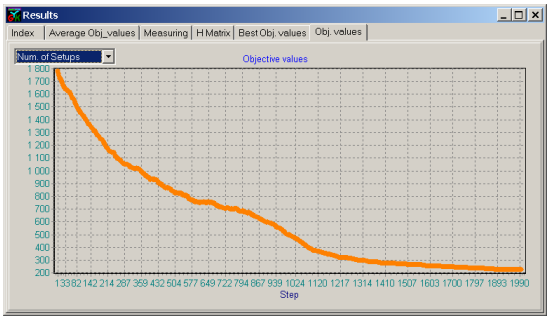

Figure 9. Number of Setups (Set in li1 test)

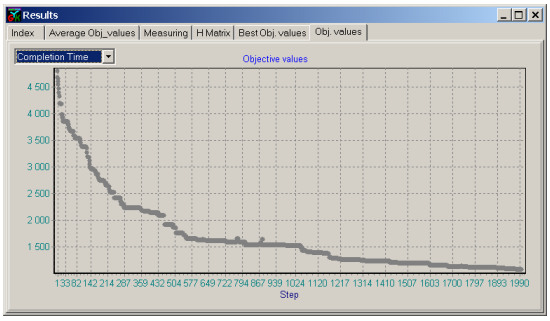

Figure 10. Maximum Completion Time (maxC in li1 test)

VII. Conclusions

This paper described the proposition and application of a new method for multi-objective scheduling and rescheduling problems. It is based on new interpretation and usage of relational operators for schedules in searching algorithms. After developing computer program, the concept is successfully tested on extended flexible flow shop scheduling and rescheduling problems (originated from an industrial project) under multiple objectives.
The obtained results and the problem independent nature of the approach are encouraging to the application of the method in other multi-objective optimization problems.

Acknowledgments

The result of research and development summarized in this paper are 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 express their thanks for the support.

References

[1] A. Baykasoğlu, L. Özbakir, T. Dereli, "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.

[2] D. Quadt, H. Kuhn. "A Taxonomy of Flexible Flow Line Scheduling Procedures", *European Journal of Operational Research,* 178, pp. 686-698, 2007,

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

[4] G. Vieira, J. Hermann, E. Lin, "Rescheduling Manufacturing Systams: A Framework of Strategies, Policies and Methods", *Journal of Scheduling*, 6 (1), pp. 35-58, 2003.

[5] H. Aytug, M. A. Lawley, K, Mckay, S. Mohan, R. Uzsoy. "Executing Production Schedules in the Face of Uncertainties: A Review and some Future Directions", *European Journal of Operational Research*, 161, pp. 86-110, 2005.

[6] K. L. Smith, R. M. Everson, J. E. Fieldsend, "Dominance Measures for Multi-Objective Simulated Annealing". In *Proceedings of Congress on Evolutionary Computation*, pp. 23-30, 2004.

[7] K. N. Mckay, V. C. Wiers, "Unifying the Theory and Practice of Production Scheduling", *Journal of Manufacturing Systems*, 18 (4), pp. 241-248, 1999.

[8] L. F. Sbalzarini, S. Müller, P. Koumoutsakos. "Multiobjective Optimization Using Evolutionary Algorithms". In *Center of Turbulence Research, Proceedings of the Summer Program 2000*, pp. 63-74, 2000.

[9] R. Linn, W. Zhang. "Hybrid Flow Shop Scheduling: A Survey", *Computers and Industrial Engineering*, 37, (1-2), pp. 57-61, 1999.

[10] T. Kis, E. Pesch. "A Review of Exact Solution Methods for the Non-Preemptive Multiprocessor Flowshop Problem", *European Journal of Operational Research*, 164, (3), pp. 573-695, 2005.

[11] T. Loukil, J. Teghem, D. Tuyttens, "Solving Multi-Objective Production Scheduling Problems Using Metaheuristics", *European Journal of Operational Research*, 161, pp. 42-61, 2005.

[12] T. Yamada. "Studies on Metaheuristics for Jobshop and Flowshop Scheduling Problems". *Ph.D Thesis*, Kyoto University, Japan, 2003.

[13] W. Wang. "Flexible Flow Shop Scheduling: Optimum, Heuristics, and Artificial Intelligence Solutions", *Expert Systems*, 22, (2), pp. 78-85, 2005.

[14] X. Zhu, W. E. Wilhelm. "Scheduling and Lot Sizing with Sequence-Dependent Setup: A Literature review", *IIE Transactions*, 38, pp. 987-1007, 2006.

Author Biographies

Gyula Kulcsár was born in 1978 in Kazincbarcika, Hungary. He received the M.Sc. Degree in information engineering from the Faculty of Mechanical Engineering at the University of Miskolc, Hungary, in 2001. He studied in József Hatvany Doctoral School of Information Science, Engineering and Technology at University of Miskolc. Since 2004, he has been an assistant professor at the Department of Information Engineering at the University of Miskolc. His research interests include production scheduling/rescheduling, multi-objective optimization, manufacturing execution systems and software engineering.

Ferenc Erdélyi was born in 1932 in Debrecen, Hungary. He graduated in 1956 from the Faculty of Mechanical Engineering at the Technical University of Miskolc. He earned a second degree in 1964 from the Faculty of Electrical Engineering at the Technical University of Budapest. He earned his CSc in 1993 from the Hungarian Academy of Science. At present he is a retired associate professor at the Institute of Information Technology at the University of Miskolc and senior researcher at the Technical University of Budapest. He is a member of the Hungarian Engineering Academy. His major research fields are the IT for manufacturing applications, and the modeling and control of manufacturing systems.