Production Systems and Information Engineering
Volume 10 (1), pp. 76–88
doi: 10.32968/psaie.2022.1.7.

AN ADVANCED MODEL FOR SOLVING INDUSTRIAL SCHEDULING PROBLEMS

Gyula Kulcsár
University of Miskolc, Hungary
Department of Information Engineering
iitkgy@uni-miskolc.hu

Mónika Kulcsárné Forrai
University of Miskolc, Hungary
Department of Information Engineering
aitkfm@uni-miskolc.hu

Abstract. This paper summarizes an advanced model and a practice-oriented approach to solve scheduling problems of discrete manufacturing systems. An advanced scheduling approach is presented which is able to adapt to the requirements of real-life situations by taking into consideration the specific characteristics of modern manufacturing and assembly systems. These detailed constraints and capabilities of the actual resource environment include the alternative technological routes, the limited available machines, the unrelated processing time, the sequence dependent setup time and the jobs with due dates. An extended flexible job shop scheduling model is defined to solve the resource allocation problems and to create the fine schedule of the execution of jobs, tasks and operations. The paper describes the most important characteristics of the analyzed problem class and shows the main approach of the developed heuristic solving method. Our approach combines a special searching technique based on fast execution-driven simulation with a multi-objective optimization model.

Keywords: Scheduling, Manufacturing, Simulation, Search algorithm, Optimization

1. Introduction

In modern manufacturing and assembly environments, many scheduling problems occur. Production scheduling can be defined as the allocation of available production resources over time to perform a collection of tasks [1] . Most of the scheduling problems are highly complicated to solve. The background is that scheduling problems lead to combinatorial optimization tasks.

This entanglement stems from their nature. Nowadays, the production planners, engineers and managers utilize computer integrated application systems to support decision making [2]. They can use a generic spreadsheet applications, ERP-integrated services, external APSs, or other software solutions to solve scheduling problems.

This paper is primarily concerned with industrial scheduling problems, where it is required that advanced scheduler software has to assign limited available resources to the operations 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. A series can include one or more pieces. The discrete manufacturing operations are executed on discrete, separated machines and workplaces. A wide variety of discrete manufacturing systems can be developed depending on the arrangement of the system components (for example machines, workstations, robots, buffers, material handling devices and so on), Based on the logical structures, technological rules and alternatives for performing operations, the manufacturing systems can be characterized by different resource environments (i.e.: single machine, parallel machines, flow shop, flexible flow shop, job shop, flexible job shop and so on). In this paper we are dealing with the scheduling problems in which the execution of the operations requires the exact predefinition of the feasible routing alternatives.

The focus of this paper is set to the fine (detailed) scheduling function of the shop floor control systems. The main purpose of the fine scheduler software is to create a detailed and near-optimal schedule to meet the master plan defined at the Enterprise Resource Planning (ERP) level. The scheduler collects and reads the actual input data of dependent (internal) production orders, product types, available resource environment, technological process plans and additional constraints and restrictions. The shop floor management configures the actual production goals and their priorities. Obviously, management may declare various goals time by time. The scheduler software must create a feasible schedule which meets the management's goals. The result of scheduling is a fine schedule which provides the releasing sequence, starting, processing, and finishing time data of the jobs and their operations. The schedule assigns the concrete resources to each activity. The execution of the fine schedule must meet the pre-defined goals without breaking any of the hard constraints. In addition to the efficiency of the result schedule, the running time of the scheduling process is also an important issue especially with large number of internal orders, jobs, operations, resources, technological variants and constraints.

2. Literature Review

In the literature, different flexible scheduling model variants can be found. One of the main groups of these models is the flexible flow shop (FFS) scheme. A detailed survey for the FFS problem is given by Quadt and Kuhn [3]. FFS environment consists of stages that represent the fundamental (operation-type) units 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 machines. The sequence of the visited stages is given.

Another class of flexible scheduling models is the flexible job shop (FJS). This problem type is an extension of the classical job shop scheduling problem. In FJS an operation can be assigned to any member of the given set of eligible machines during scheduling [4]. The problem is to assign each operation to a chosen suitable machine and to create the execution sequence of the operations on each machine such that the objective function will be optimized or near-optimal. The flexible flow shop problem is a special case of the flexible job shop problem.

The allocation of machines and the sequence of jobs strongly influence the quality of the schedule. The production performance depends on both factors. Many shop scheduling models can be found in the literature, but most of them use only one performance measure. Usually the makespan (latest finishing time of the released jobs) appears as objective function of optimization for Make to Stock (MTS) manufacturing. Objective functions related to due date play the main role in scheduling for supporting make to order (MTO) manufacturing. In such cases, for example, the maximum tardiness, the sum of tardiness and the number of tardy jobs are frequently used as objective functions.

The number of single-objective scheduling models is much larger than the number of multi-objective scheduling models which are more important in flexible and agile manufacturing. For example, references [5] and [6] show two different approaches. Moreover, the existing models often disregard the machine processing abilities, alternative process plans, limited machine availability time frames, job travelling times, and machine eligibility. Although scheduling models are rapidly expanding and evolving, industrial production systems are generating more and more problems, then the improvement and extension of flexible shop floor scheduling models are required.

3. An Advanced Scheduling Model and new Algorithms

3.1. The Goal of the Research

The starting basis of our research was the Extended Flexible Flow Shop model (EFFS) [7]. Developing a new model for flexible manufacturing systems was the main goal of our research. We have defined a new model, namely the Extended Flexible Job Shop (EFJS), in order to extend the set of manageable operation execution characteristics and alternative technological process plans.

3.2. An Extended Flexible Job Shop Model

From the operation execution point of view, in the job shop (JS) resource environment, there exists only one available machine for each operation and one feasible process plan for each job. In the flexible job shop (FJS) model, there are choices to be made in solving the scheduling problem from among alternative parallel machines (set of suitable machines) on which an operation can be performed. In our new EFJS model, we assume that alternative technological process plans can be assigned to each job. The process plan specifies the number and sequence of operations to be performed. The EFJS scheduling problems is more complicated decision (optimization) problem than FJS problem. As the well-known JS and FJS are NP-hard combinatorial optimization problems, therefore our EFJS scheduling problem is also NP-hard.

In the EFJS model, the discrete manufacturing system produces various products. By means of forecasting tools which consider external orders, market trends, seasonal characteristics, an actual set of internal production orders has been created by the production planners. Each production order defines the required number of identical products of certain product type, which should be manufactured by the pre-defined due date. The logistical unit is the palette at the shop floor level, which can take one or more products. An internal production order consists of one or more palettes. Depending on the product type, palettes carry pre-defined number of identical products. A given production order (PO) can be considered as the set of palettes to be executed, where the number of palettes depends on the ordered product quantity and the capacity of the palettes. Our model applies manufacturing/assembly machine/workstation objects (individual machines, flexible machine/assembly lines, and flexible manufacturing cells). Machine/assembly lines and manufacturing cells can perform more than one technological steps (TS). Each TS means a sequence of operations and cannot be interrupted. Consequently, TS is the smallest allocation unit of the scheduling problem. A job means one or more palette of an internal production order with technological steps to be executed in a pre-defined sequence. The nature of the extended flexible manufacturing system is that the given product type can be manufactured by using alternative materials, components, machines and technological routes (process plans). We assume that the capacity of the buffers placed among machines is not limited.

In the EFJS model, every machine object can be characterized by product sequence dependent setup times, availability time frames (calendar elements), various production rates depending on product types, and capability for performing a single TS or a sequence of TSs for certain products. The machines can be arranged into machine groups (stages) according to the processing ability. A machine group is a set of machines that can execute the same execution step (sequence of TSs). This point of view, a given final product or a given semi-finished product can be produced differently using different chains of machine groups on which the required components are taken through. The machine groups traversed by the workpiece form the technological path (route).

In order to formulate the new class of scheduling problems described above, the well-known classification scheme $\alpha|\beta|\gamma$ is used, where α denotes machine environment, β denotes processing characteristics and constraints, and γ denotes the list of objective functions. We define the Extended Flexible Job Shop (EFFS) scheduling model as follows:

$$EFJS, M_g, Q_{i,m}, Set_{i,j,m}, Cal_m | r_i, d_i, Exe_i, A_{i,g} | f_1, f_2, \ldots, f_K \tag{3.1}$$

where the symbols are as follows:

EFJS – extended flexible job shop,

M_g – group of multi-purpose parallel machines,

$Q_{i,m}$ – unrelated parallel machines with job dependent production rates,

$Set_{i,j,m}$ – job sequence and machine dependent setup times,

Cal_m – machine dependent availability time intervals,

r_i – job dependent release time,

d_i – job dependent due date,

Exe_i – required type and sequence of technology steps for jobs,

$A_{i,g}$ – available set of suitable machines in groups for jobs,

$f_1, f_2, \ldots, f_K$ – objective functions to be minimized (components of multi-objective optimization).

The numerical result of the objective function expresses the quality of the generated solutions (fine schedules). Some examples of typical objective functions are given in Table 1.

In real manufacturing environments, the production management may require various objective functions, therefore we focus on general multi-objective scheduling approaches in order to obtain flexible and adaptive methods for supporting shop floor scheduling in practice.

Table 1. Typical objective functions for multi-objective scheduling

Symbol	**Meaning**
C_{max}	Completion time of last job (makespan) to be min.
L_{max}	Max lateness to be min.
T_{max}	Max tardiness to be min.
S_{max}	Max square distance of differences to due dates to be min.
$\sum(C_i - r_i)$	Sum of throughput times to be min.
$\sum L_i$	Sum of lateness times to be min.
$\sum T_i$	Sum of tardiness times to be min.
$\sum U_i$	Number of tardiness to be min.
$\sum W_m$	Sum of machine blocking time to be min. (Utilization to be max.)
$\sum v_i(C_i - r_i)$	Weighted sum of flow times to be min.
$\sum v_i L_i$	Weighted sum of lateness times to be min.
$\sum v_i T_i$	Weighted sum of tardiness times to be min.
N_{WIP}	Average number of works in progress to be min.
N_{SET}	Number of setups to be min.
$\sum T_{SET}$	Sum of setup times to be min.

3.3. A new Multi-Objective Scheduling Method

The EFJS problem is difficult to solve. This model inherits the difficulties of the classical job shop and the flexible job shop models. Additionally, numerous strange features appear because of our special extensions. To solve the scheduling problems, we developed a new approach and implemented advanced algorithms.

The solution of a general, production scheduling problem includes batching (lot-sizeing), machine selection (assigning), job sequencing and operation timing decision-making sub-problems because of complexity of the entire problem. In this paper, we propose an integrated approach to solve all these sub-problems as a whole without decomposition. In this approach, all the decision-making aspects are handled simultaneously.

For solving production scheduling problems in practice, an advanced multi-objective scheduling approach has been developed based on fast production simulation [8]. After developing scheduler software, the concept is successfully tested on extended flexible flow shop problems considering multiple objectives and constraints originating from an industrial environment [9], [10]. Scheduling based on simulation can consider exactly what the actual manufacturing system should perform in the planned time horizon. In this approach, we are searching in the space of the feasible schedules such a way that each candidate schedule, which are created iteratively for the shop, meets all the hard constraints.

We overdeveloped our previous approach, which are developed for solving the EFJS problems. In the advanced approach the job plays the role of the basic scheduling unit. Each internal production order consists of jobs that mean individual unit. The unit includes (one or more workpieces with the required execution steps). To create a detailed schedule for the EFJS production system, it is necessary for each job: (1) to assign it to one of the suitable routes (technological process plans), (2) to assign it to one of the suitable concrete machines at each possible machine group according to the chosen route, (3) to fix its position in the queue of each chosen concrete machine, (4) and to determine its starting time on each chosen machine.

To make decisions in the last issue (4) is very complicated. The main idea of our approach is a problem space transformation based on a fast production simulation algorithm. We use the sequence of job-machine assignments on each machine to represent a candidate schedule as a solution. The decision variables of this reduced problem space form a simple schedule which will be extended to a fine schedule by using simulation. The simulation answers the remaining questions concerning the starting times of the execution steps (activities). Consequently, the simple schedule determines exactly the fine schedule. The simulation is implemented as an execution-driven process, in which the jobs move alone on the shop environment and have got owner time data. The sizes of the actual series (production batches) are formed dynamically by scheduling and executing the jobs on machines. Before scheduling a builder method creates the full indexed data model which provides the valid technological and resource constraints and possible alternatives.

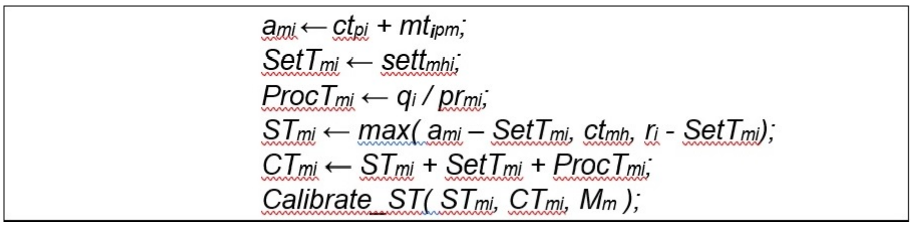
Our execution-driven simulation method can be realized with rule-based numerical simulation of the production to calculate the time data of the execution steps. Input data determine the production order, the jobs, the machines and the schedule to be executed. The schedule specifies the assignment of jobs and machines, and the schedule also defines the execution sequences of jobs on machines.

Each job is represented by an individual moving object (J_i) in the simulation. The route and the machines are chosen by the scheduler and defined in the schedule to be executed. An execution step of a given J_i on a machine means a processing task. The main steps of the simulation are as follows: (1) Build and initialize the model objects. (2) Choose the next execution step (task) to be performed. (3) Simulate the execution of the active job on the active machine. 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 influence the execution. To answer these questions the simulation must play 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 jobs. Each machine has a loading list and a pointer that shows the next job to be processed according to the schedule. The pair of a machine and its pointed job means a candidate execution step for processing if all the starting requirements are satisfied. These are as follows: (1) the machine has finished its previous job; (2) the job execution has been completed successfully on its previous machine.

The method chooses the candidate execution step which can be started earliest. The machine and the job associated with the chosen execution step become active entities. The method calculates the time data (start time ST_{mi}, setup time $SetT_{mi}$, processing time $ProcT_{mi}$, and completion time CT_{mi}) of the active job on the active machine. The processing time ($ProcT_{mi}$) is determined by the work-piece quantity (q_i) of the job, the job (product type) dependent production rate (pr_{mi}) of the machine. The start time ST_{mi} of a given job J_i on an assigned machine M_m is determined by the following values: (1) the earliest release time of the job (r_i), (2) the completion time of the job on the previous machine (ct_{pi}), (3) the moving time of the job from the previous machine (mt_{ipm}), (4) the completion time of the previous job on the machine (ct_{mh}), (5) the job-sequence dependent setup time on the machine ($sett_{mhi}$), (6) the availability time frames of the machine (CAL_m). Focusing on the simulation of the execution step of job J_i on the assigned machine M_m, the simplified description of the calculation can be seen in Figure 1 assuming that the setup activity can be started on the machine before the job arrives (a_{mi}).

The function Calibrate_ST loads the time frame required by J_i job on M_m machine. This allocation method inserts the planned 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 into the candidate available 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.

The simulation extends the pre-defined input schedule to a fine schedule by calculating and assigning the time data of the production activities. The



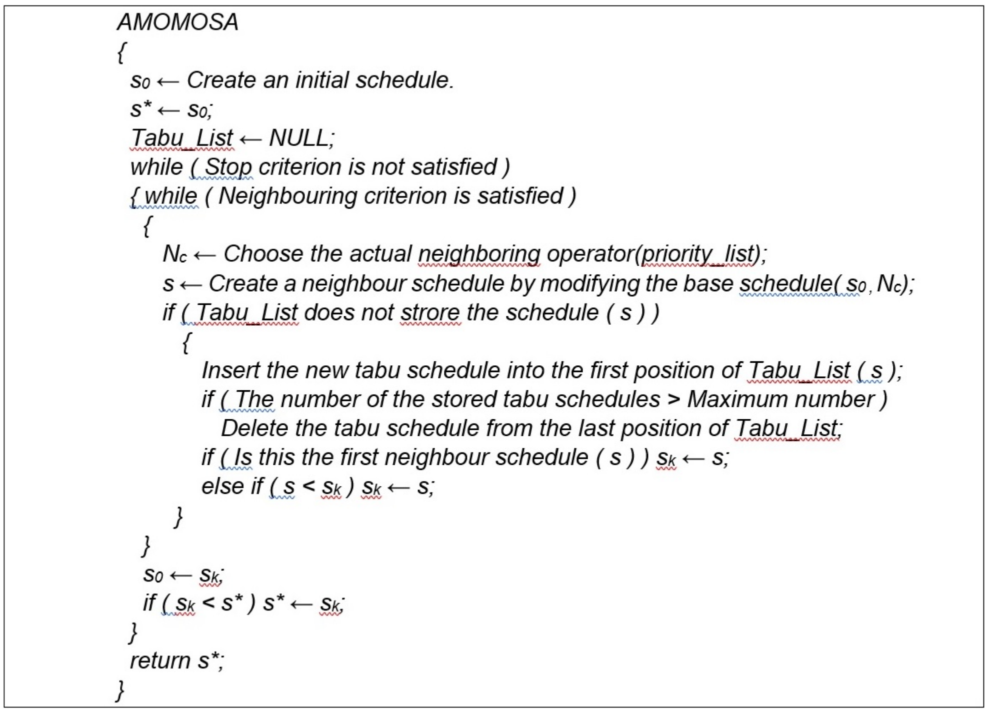
```
$a_{mi} \leftarrow ct_{pi} + mt_{ipm}$;
$SetT_{mi} \leftarrow sett_{mhi}$;
$ProcT_{mi} \leftarrow q_i / pr_{mi}$;
$ST_{mi} \leftarrow max( a_{mi} - SetT_{mi}, ct_{mh}, r_i - SetT_{mi})$;
$CT_{mi} \leftarrow ST_{mi} + SetT_{mi} + ProcT_{mi}$;
$Calibrate\_ST( ST_{mi}, CT_{mi}, M_m )$;
```

Figure 1. A simplified calculation of the time data of a given execution step

performance of the fine schedule can be measured by calculating objective functions based on the data of tasks, jobs, production orders and machines provided by the simulation. In this way the simulation transforms the original searching space of the scheduling problem to a reduced space.

The core of our scheduler software uses an advanced multi-objective and multi-operator searching algorithm (AMOMOSA). The main frame of this searching algorithm is show in Figure 2. AMOMOSA moves iteratively in the virtual space of the feasible solutions from an actual base schedule s_0 to a candidate neighbour schedule s in the neighbourhood of s_0 until the stop criterion is satisfied. To reach and examine the unexplored regions of the searching space the AMOMOSA modifies the neighbourhood structure of each base schedule as the searching progresses. To escape local optimum the AMOMOSA stores the schedules that have been visited in the recent past in a tabu list. The maximum number of elements stored in the list is pre-defined by a parameter. The schedules, which are stored in the tabu list, are excluded from the actual neighbourhood of the actual base schedule. New neighbours of the current base schedule are created randomly by neighbouring operators. These operators generate candidate schedules by modifying the resource allocations, the job sequences according to the problem space characteristics. These are the primary decision variables of the scheduling problem.

The proposed neighbouring operators are as follows: N_1 operator removes a production order randomly chosen from the schedule then inserts all the jobs of the production order as a matching series. These jobs travel the same route and visit the same machines. On the selected machines, these jobs are performed in a given sequence. N_2 operator removes a late production order randomly chosen from the schedule then inserts all the jobs of the production order Inserts jobs just like the N_1 operator. N_3 operator modifies the sequence of jobs on a randomly chosen machine by using random length permutation-cycle. N_4 operator removes a late job randomly chosen from the schedule then redefined the manufacturing tasks of the job by assigning resources and finally



```
AMOMOSA
{
  $s_0 \leftarrow$ Create an initial schedule.
  $s^* \leftarrow s_0$;
  Tabu_List $\leftarrow$ NULL;
  while ( Stop criterion is not satisfied )
  { while ( Neighbouring criterion is satisfied )
    {
      $N_c \leftarrow$ Choose the actual neighboring operator(priority_list);
      $s \leftarrow$ Create a neighbour schedule by modifying the base schedule( $s_0$, $N_c$);
      if ( Tabu_List does not strore the schedule ( s ) )
        {
          Insert the new tabu schedule into the first position of Tabu_List ( s );
          if ( The number of the stored tabu schedules > Maximum number )
            Delete the tabu schedule from the last position of Tabu_List;
          if ( Is this the first neighbour schedule ( s ) ) $s_k \leftarrow s$;
          else if ( $s < s_k$ ) $s_k \leftarrow s$;
        }
    }
    $s_0 \leftarrow s_k$;
    if ( $s_k < s^*$ ) $s^* \leftarrow s_k$;
  }
  return s*;
}
```

Figure 2. An Advanced Multi-Objective and Multi-Operator Search Algorithm (AMOMOSA)

inserts the job into random position on each related machine. N_5 operator works similarly to the N_4 operator, but the target job is chosen from the set of all jobs.

These neighbouring operators create new candidate schedules by modifying the values of the primary decision variables of the base schedule. Our searching algorithm synthesizes the advantages of the variable neighbourhood structure and the tabu search.

The return values at 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. The precise definition of the comparing operator is given in [7], [8] and [9].

For managing effectively all the objective functions of the multi-objective optimization, we used our own previously elaborated method. Our model is based on the calculation of the relative quality of a given solution by comparing it to another solution, considering multiple objective functions simultaneously.

The formal description of the relative qualification model is as follows:

$$f_k : S \to \mathbb{R}^+ \cup \{0\}, \forall k \in \{1, 2, \dots K\}, f_k \to min \tag{3.2}$$

$$D : \mathbb{R}^2 \to \mathbb{R}, D(a,b) = \begin{cases} 0 & \text{, if max(a,b)=0,} \\ \frac{b-a}{max(a,b)} & \text{, otherwise.} \end{cases} \tag{3.3}$$

$$F : S^2 \to \mathbb{R}, F(s_x, s_y) = \sum_{k=1}^{K} (w_k D(f_k(s_x), f_k(s_y))), \tag{3.4}$$

where

$S-$ the set of the feasible solutions;

f_k- the k^{th} objective function to be minimized;

$K-$ the number of objective functions;

s_x, s_y- two given solutions;

w_k- the priority of the k^{th} objective function;

$F(s_x, s_y)-$ the relative quality of the solution s_y compared to the solution s_x.

Using the signed return value of the function $F(s_x, s_y)$ we extend the interpretation of the relational operators to the solutions s_x and s_y in the S. The definition of this operator overloading is the following:

$$(s_y ? s_x) := (F(s_x, s_y) ? 0) \tag{3.5}$$

where the question mark ? indicates any of the relational operators $(<, \leq, >, \geq, =, \neq)$.

The relationship between $F(s_x, s_y)$ and zero expresses the relationship between solution s_y and solutions s_x. 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.

This relative qualification model can effectively solve the comparison of the candidate solutions in the searching algorithm, so the proposed AMOMOSA method is able to realize multi-objective optimization by taking into account the actual requirements of the production management.

4. Conclusion

In this paper, we focused on modelling of multi-objective scheduling problems originated from discrete production systems, and we presented new methods to create near-optimal feasible schedules considering detailed constraints and capabilities of the extended flexible job shop. It is a very important and complicated task to make efficient schedules for the shop floor to manage different types of resources and many operations.

The proposed model supports the flexible and effective usage of production management goals and requirements simultaneously. We presented the integrated fine scheduling approach that supports the optimization of planning the internal production orders; the calculation of the manufacturing batch sizes dynamically; the loading of the alternative technological routes; the allocation of machines, workstations and other resources; the definition of manufacturing tasks and the scheduling of the execution processes and activities. The proposed method uses an advanced multi-objective multi-operator searching techniques and problem space transformation based on a discrete execution-driven simulation. The proposed simulation provides the concrete time data for the manufacturing processes and activities. The simulation extends the candidate schedule to a fine schedule by calculating the starting and finishing time data of all activities. Consequently, the simulation is able to transform the original searching space to a reduced space by solving the timing sub-problem of the entire scheduling problem. This feature of our approach encapsulates the dependency of industrial scheduling problems. Therefore, the proposed model and method are widely applicable in the industry.

References

[1] Baker, K.: *Introduction to Sequencing and Scheduling*. Wiley, 1974, ISBN 9780471045557, URL https://books.google.hu/books?id=MslTAAAAMAAJ.

[2] Kulcsár, G., Hornyák, O., and Erdélyi, F.: Shop floor control decision supporting and mes functions in customized mass production. *Manufacturing Systems Development–Industry Expectation*.

[3] Quadt, D. and Kuhn, H.: A taxonomy of flexible flow line scheduling procedures. *European Journal of Operational Research*, **178**(3), (2007), 686–698, URL https://doi.org/10.1016/j.ejor.2006.01.042.

[4] Defersha, F. M. and Chen, M.: A parallel genetic algorithm for a flexible job-shop scheduling problem with sequence dependent setups. *The International Journal of Advanced Manufacturing Technology*, **49**(1), (2010), 263–279, URL https://doi.org/10.1007/s00170-009-2388-x.

[5] Loukil, T., Teghem, J., and Tuyttens, D.: Solving multi-objective production scheduling problems using metaheuristics. *European Journal of Operational Research*, **161**(1), (2005), 42–61, URL https://doi.org/10.1016/j.ejor.2003.08.029.

[6] Smith, K. I., Everson, R. M., and Fieldsend, J. E.: Dominance measures for multi-objective simulated annealing. In *Proceedings of the 2004 congress on*

evolutionary computation (IEEE Cat. No. 04TH8753), vol. 1, IEEE, 2004, pp. 23–30, URL https://doi.org/10.1109/CEC.2004.1330833.

[7] Kulcsár, G. and Erdélyi, F.: A new approach to solve multi-objective scheduling and rescheduling tasks. *International Journal of Computational Intelligence Research*, **3**(4), (2007), 343–351.

[8] Kulcsár, G. and Kulcsárné, F.: Solving multi-objective production scheduling problems using a new approach. *Production Systems and Information Engineering, A Publication of the University of Miskolc*, **5**, (2009), 81–94.

[9] Kulcsár, G.: A practice-oriented approach for solving production scheduling problems. In *Proceedings of the XXV microCAD International Scientific Conference, Miskolc, Hungary*, 2011, pp. 61–66.

[10] Kulcsár, G. and Forrai, M. K.: Detailed production scheduling based on multi-objective search and simulation. *Production Systems and Information Engineering*, **6**, (2013), 41–56.