LEARNING TECHNOLOGY IN SCHEDULING BASED ON THE MIXED GRAPHS

Yuri Sotskov, Nadezhda Sotskova, Leonid Rudoi

Abstract: We propose the adaptive algorithm for solving a set of similar scheduling problems using learning technology. It is devised to combine the merits of an exact algorithm based on the mixed graph model and heuristics oriented on the real-world scheduling problems. The former may ensure high quality of the solution by means of an implicit exhausting enumeration of the feasible schedules. The latter may be developed for certain type of problems using their peculiarities. The main idea of the learning technology is to produce effective (in performance measure) and efficient (in computational time) heuristics by adapting local decisions for the scheduling problems under consideration. Adaptation is realized at the stage of learning while solving a set of sample scheduling problems using a branch-and-bound algorithm and structuring knowledge using pattern recognition apparatus.

Keywords: Scheduling, mixed graph, learning, pattern recognition.

ACM Classification Keywords: F.2.2 Nonnumerical algorithms and problems: sequencing and scheduling.

Introduction

Scheduling is defined as an optimal assignment of the limited resources (machines) to competitive jobs over time. Having arisen from practical needs, scheduling theory attracts much attention of management workers and operations research practitioners. One of the most general scheduling problem may be described in a framework of the multistage system with machine set M including both different machines and identical machines that should process the given set of jobs $J = \{J_1, J_2,..., J_n\}$. Each job J_r consists of the ordered set of operations. Processing time p_i and type of machine from set M that has to process operation i are assumed to be known before scheduling. Let $M = \bigcup_{k=1}^{w} M_k$ where M_k denotes a set of identical machines of type k. If $\mu_i \in M_k$ and $\mu_j \in M_l$ with $k \neq l$, then machines μ_i and μ_j are of different types. If $k = l$, machines μ_i and μ_j are identical. We assume that operation preemptions are not allowed. As usual, each machine cannot process two (or more) operations simultaneously. The objective is to assign all the operations to suitable machines from set M, and to define for each machine $\mu_j \in M$ the sequence of operations assigned to μ_j in such a way that the value of the given objective function is minimal.

The complexity research has shown that even very special cases of the above scheduling problem (e.g., problems with either two different or two identical machines, or problem with three jobs and three different machines) are NP-hard [16]. Therefore, the use of exact scheduling algorithms is very limited in practice. The general methodology for developing exact scheduling algorithms is based on the branch-and-bound (B&B) method [4,14,16]. The exhausting enumeration while *branching* set of feasible schedules ensures the optimality of the best schedule constructed. The lower and upper *bounds* of the given objective function allow eliminate schedules, which are dominated, and reduce considerable enumeration process. However, we are forced to recognize that current improvements of the B&B algorithms cannot change radically the limit on the size of the scheduling problems solvable within reasonable running time. Moreover, it is unlikely to construct fast exact algorithms and even approximate ones with good performance measure for scheduling problems with real-world size. To be more application-oriented, different heuristics have been developed to find near-optimal feasible schedules.

It should be noted that the reasons for applying certain heuristics in dispatching algorithms are usually weakly grounded and are based mainly on the experiments. The choice of heuristics in the knowledge-based system is more adequate and it is usually computer-aided. In article [1], it is shown that a better solution can be often obtained due to appropriate changing the set of dispatching rules. As a result, the obtained processing system became better than that obtained using only a fixed set of dispatching rules. In article [12], it is demonstrated that

machine learning produces an improvement in the performance of the processing system when compared to the traditional method of using a fixed set of dispatching rules.

Local decision rules are the “backbone” for heuristic algorithms. The outcome of scheduling algorithm depends on the underlying strategy. As it was mentioned in [19], artificial intelligence provides often a better basis for modeling and solving a scheduling problem. Many real-world scheduling problems seem to be solved by semi-logical methods, such as recognizing one of a thousand familiar patterns applying to current situation and recalling the appropriate thing to do when that pattern occurs. In article [6], it was shown that by combining a learning system with simulation, a manufacturing control system can be developed that learns from its historical performance and makes its own scheduling and control decisions by simulating alternating combinations of different dispatching rules. The objective of the research described in [8] was to study the feasibility of automatically scheduling multi-machine complexes and adjusting the schedule on a real-time basis by a unified computer system. In article [2], a probabilistic learning strategy has been developed based on the principles of evolutions. In [10,17,18], it was shown how an algorithm could be extended to consider local decision rules of any number and complexity. The resulting programs do better than that based on the deterministic approach. In article [3], it is emphasized that meta-heuristics are general combinatorial optimization techniques, which are designed with the aim of being flexible enough to handle as many combinatorial problems as possible. In the recent years, these techniques have rapidly demonstrated their usefulness and efficiency in solving different combinatorial problems including scheduling problems.

Reviews of the learning-based scheduling approaches are presented in [8,11,19]. A common approach to scheduling set of the given jobs in practice is based on different dispatching rules. The performance of these rules depends on the state of the system at each moment, since there is no single rule that is better than others in all the possible system states. So, it is useful to look for the most appropriate dispatching rule at each state of the system. To achieve this goal, a scheduling approach based on machine learning seems to be rather promising. Due to analysis of the previous performance of the system, it is often possible to get knowledge that can be used to decide which dispatching rule is the most appropriate at this or that state of the system under consideration.

In this paper, exact algorithm, heuristics and pattern recognition apparatus are used in common. Our approach may be considered as a generalization of the algorithm proposed in [13] for the very special case of the above scheduling problem when all machines in set M are different (job shop). We use mixed graph model [15] along with so-called conflict resolution strategy [14–16]. The main issue of this paper is a classification scheme for local conflicts due to computational results of the exact and approximate versions of the B&B algorithm. In the next section, we formulate the scheduling problem in terms of the mixed graph model and give an overview of the conflict resolution strategy being a basis for different exact, approximate and heuristic scheduling algorithms. Modules 1, 2 and 3 of the knowledge-based system are briefly discussed in the next three sections. Some remarks are given in Conclusion.

Mixed Graph Model

A multistage system with both different and identical machines may be described via mixed graph $G = (V, A, E)$ [15], which is an appropriate model for constructing various scheduling algorithms [14,16]. In such a model, operation set V is represented as a set of vertices $V = \{1,2,..., n\}$, a weight prescribed to each vertex $i \in V$ being equal to the processing time p_i of operation i. As usual, precedence constraints between operations in set V are represented via arc set A. Let equality $V = \bigcup_{k=i}^{w} V_k$ hold, where V_k denotes the set of all operations processed by machines of type $k \in \{1, 2,..., w\}$, and let $w_k = |M_k|$ be the number of identical machines of type k provided that equality $|M| = \sum_{k=1}^{w} w_k$ holds. Competitions among operations that have to be processed by machines of the same type are represented via edge set E, namely: edge $[i, j]$ belongs to set E if and only if operations i and j belong to the same set V_k.

If $r \in V_k$ and $s \in V_l$ with $k \neq l$, then operations r and s have to be processed by different machines (namely, by machines of different types: one machine from set M_k, and another machine from set M_l). For each pair of operations i and j with edge $[i, j] \in E$ there exist three possibilities represented in Fig. 1.

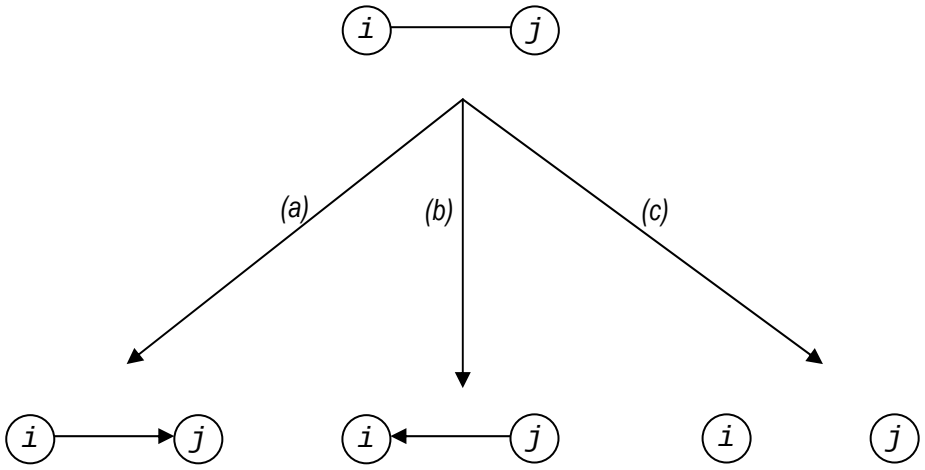

Figure 1. Case (a): $(i, j) \in A[\alpha^h]$ (
Case (b): $(i, j) \in A[\alpha^h]$, respectively) if operations i and j are assigned to the same machine, and operation i (operation j) is processed before operation j (operation i);
Case (c): edge $[i, j]$ is removed from the mixed graph G if operations i and j are assigned to two different machines of the same type

Therefore, assigning operations V to machines M and sequencing operations assigned to the same machine correspond to orienting or removing edges from set E in one of these three possibilities. If all such edge transformations do not cause conflicts in the resulting digraph, then they define assigning operations V_k to machines M_k, $k = 1, 2,..., w$, and define the sequence of operations assigned to each machine $\mu_u \in M$. Acyclic digraph constructed in such a way corresponds to a feasible schedule, if the number w_k of the available machines of each type k is sufficient for such a transformation of edges in set E. More precisely, selecting edges $[i, j]$ one by one from set E of the mixed graph G, and applying to each edge $[i, j]$ one of the three possible transformations (removing edge $[i,j]$, or substituting it either by arc (i,j) or by arc (j,i)), we obtain transformation sequence $\alpha^h = (\alpha^h_1, \alpha^h_2, ..., \alpha^h_{|E|})$ of all the edges in the mixed graph G. Let $G[\alpha^h] = (V, A[\alpha^h], \varnothing)$ denote a digraph obtained from the mixed graph G as a result of the above transformation α^h. In article [15], the following claim has been proven.

Theorem. *Digraph $G[\alpha^h]$ defines a feasible schedule if and only if the following three conditions hold:*

1) *digraph $G[\alpha^h]$ contains no circuits;*
2) *for any $k \in \{1, 2,..., w\}$, the number of components in subgraph $(V_k, A_k[\alpha^h], \varnothing)$ of digraph $G[\alpha^h]$ is not greater than w_k;*
3) *for any $k \in \{1, 2,..., w\}$, each component of the digraph $(V_k, A_k[\alpha^h], \varnothing)$ is a tournament.*

Let $P(G)$ denote the set of all digraphs $G[\alpha^h]$ satisfying all conditions 1), 2) and 3) given in Theorem. To find optimal schedule we may enumerate (e.g., implicitly via B&B method) digraphs from set $P(G)$, and choose the *optimal digraph,* i.e., that with minimal value of the given objective function.

Edge $[i, j] \in E$ is called *conflict* if both its orientations (i,j) and (j, i) in the mixed graph G cause an increase of the starting time of at least one operation from set V. Using *conflict resolution strategy,* we deal with one conflict edge at each iteration of the B&B algorithm and test three possible transformations of this edge. The rule for choosing the edge from the set of conflict ones may be different, but the common idea is to choose that with maximal value of the conflict measure [4,14,15]. To find a solution of the scheduling problem, an *exact* B&B algorithm may be used to realize implicit enumeration of all feasible schedules. An *approximate version* of a B&B algorithm may be obtained from exact one using some error bound allowed for the desired objective function value. A *heuristic*

algorithm may be constructed by considering only one branch of the solution tree via choosing at each iteration single transformation of the conflict edge (e.g., due to some priority rules used for choosing operation from the set of two operations involved in conflict). Thus, mixed graph model with conflict resolution strategy may be a foundation for constructing a variety of scheduling algorithms differing in the running time and in accuracy of the best solution obtained.

In the learning technology, we use the exact (or approximate version of) B&B algorithm in Module 1, while in Module 3 we use adaptive algorithm with logical rules produced within Module 2 by inductive inference. In the following sections, we show how to realize such an adaptation using mixed graph *G.*

Stage of Learning (Module 1 and Module 2)

At the stage of acquiring knowledge, we accumulate the knowledge about scheduling problems under consideration. Module 1 solves a set of sample scheduling problems via exact B&B algorithm (or via approximate B&B algorithm with allowed upper bound of the objective function) and stores information on the successful local decisions that have led to the optimal schedule (or to the best schedule obtained). The main idea for acquiring knowledge is to accumulate (in so-called *learning table*) the information that was obtained during the use of the time-consuming exact (or approximate version of) B&B algorithm. Next, we show how this knowledge could be used when solving other scheduling problems that are close to previous ones already solved by Module 1 (i.e., problems with similar structure of the processing system, close numerical data, the same objective function, etc.).

The aim for constructing learning table is to accumulate the knowledge on successful local decisions used in the historical performance of the B&B algorithm. The stage of learning consists in accumulating information and tuning some parameters of the B&B algorithm to the properties of a set of scheduling problems to be solved. The learning table is analogous to those used in *pattern recognition.* It describes which transformation of a conflict edge seems to be preferable. While solving the sample problem, information on successful transformations of the conflict edges has to be written in the learning table (see Table 1 without last row). Each raw of this table corresponds to one conflict edge. After the B&B algorithm from [15] is stopped, the path in the solution tree is known that has led to the optimal schedule (or to the best schedule obtained). We can consider the set of conflict edges tested in this path. If the B&B algorithm chooses the most conflict edge $[i_1,j_1]$ as the first one, then Module 1 considers edge $[i_1,j_1]$ as the first one too, and so on. Columns in Table 1 corresponding to characteristics are filled by calculated characteristic values x^i of the object (conflict edge) $[i_m,j_m]$. The last column in Table 1 contains classes Ω_1, Ω_2,..., Ω_7 attributed to objects $[i_m,j_m]$. Let $T_{u,v}$ denote the learning table with v objects and u characteristics. In our software, number u is the parameter of the program fixed at the initial step of the algorithm, while number v of the objects may grow with the growth of the number of sample problems solved by Module 1.

Table 1: The learning (recognition) table $T_{u,v}$ used in Module 2 (Module 3)

Objects	Characteristic values				Classes of objects
(conflict edges)	x^1	x^2	....	x^u	$\Omega_1, \Omega_2,..., \Omega_7$
$[i_1, j_1]$	b^1_1	b^2_1	…	b^u_1	Ω_{k1}
$[i_2, j_2]$	b^1_2	b^2_2	…	b^u_2	Ω_{k2}
…	…	…	…	…	…
$[i_v, j_v]$	b^1_v	b^2_v	…	b^u_v	Ω_{kv}
$[i, j]$	b^1	b^2	…	b^u	?

Notion of characteristic like x^i is important for object description in any knowledge-based system. Next, we demonstrate how characteristics of the conflict edges may be introduced by generalizing priority rules. While creating the learning table for the first time, the characteristics are chosen from the known list, and they are ordered with respect to their theoretical informative measure. Each characteristic may correspond to a priority rule

for a conflict resolution. After words having adopted on the set of scheduling problems, Modules 1 and 3 may also use more complex and more informative logical rule as a new characteristic created by Module 2. In this case, useless (non-informative) characteristic x^u has to be rejected from table $T_{u,v}$ (since number u of characteristics used in table $T_{u,v}$ is fixed at the initial step of the algorithm). Index m from set {1, 2,…, u} and a position in the ordered set of characteristics have to be prescribed to the new characteristic with respect to its usefulness: the more informative characteristic is, the smaller index m it must have. Such a replacement of characteristics is realized within Module 2. In the experiments, we used first the characteristics corresponding to the priority rules given in [9]. A priority rule allows compare priority values of two operations competing on machines of the same type. It is useful to introduce the corresponding characteristic equal to the difference between these priority values for two operations i and j for the conflict edge [i,j]. To make such characteristics more universal (and so applicable to other scheduling problems), it is reasonable to use the relative difference of these priority values obtained after dividing them by a common value (e.g., by the maximal value of this characteristic in the whole mixed graph G). Next, we demonstrate typical examples of characteristics used in computational experiments.

SPT rule recommends to process first the operation with the "Shorter Processing Time", and then the operation with the greater processing time. So, we can introduce corresponding characteristic as follows:

$$x_1 = (p_i - p_j) / \max\{p_k : k \in V\}.$$

Value x_1 characterizes edge [i, j] more adequate than the corresponding priority rule: the sign of this value shows which operation i or j has the shorter processing time, and absolute value of x_1 makes it possible to compare edges of the mixed graph G with those of the mixed graphs constructed for other problems.

FIFO rule recommends to process first the operation that has the shorter starting time ("First In, First Out"). The starting time s_i of operation i can be easily calculated as the maximal weight of the path ending in vertex i of the digraph (V, A, $\varnothing$). Value x_2 of a characteristic corresponding to priority rule *FIFO* may be calculated for edge [i, j] as follows:

$$x_2 = (s_i - s_j) / \max\{s_k : k \in V\}.$$

In the computational studies (realized for the special case of the problem with $|M_k| = 1$ for each $k = 1, 2,\ldots,w$) especial attention has been paid to priority rule *LIOF* ("Least Increase in the Objective Function"). Considering two operations i and j processed by the same machine, two values F_{ij} and F_{ji} of the objective function F are calculated for two possible orders to process these operations. Value F_{ij} is calculated for partial schedule with operation i being processed first and operation j being processed second (in this case, conflict edge [i, j] has to be substituted by arc (i,j)). Value F_{ji} is calculated for partial schedule with operation j being processed first and operation i being processed second (edge [i, j] has to be substituted by arc (j, i)). The *LIOF* rule recommends to choose the order defined by arc (i,j) for operations i and j if $F_{ij} < F_{ji}$, and to choose the opposite order defined by arc (j, i), otherwise.

The computational studies with three famous job shop problems from [7] showed that *LIOF* rule is very informative. More precisely, it was shown (by experiments on computer) that while constructing an optimal schedule for the makespan criterion $C_{max} = \max\{C_i : i \in V\}$ (where C_i means the completion time of job $J_i \in J$) via Module 1, the most part of the conflict edges was successfully oriented just due to the *LIOF* rule. For test problems, number of edges substituted by arcs with respect to *LIOF* rule formed about 60 percent, and the number of exceptions formed often only 1 percent of the whole number of conflict edges tested in Module 1. For the rest 39 percent of conflict edges [i,j], operations i and j were often equivalent in the sense of equality $F_{ij} = F_{ji}$.

Thus, at the learning stage, we consider the conflict edge of the mixed graph G as an *object* (using pattern recognition term). All columns in table $T_{u,v}$ (except the first and the last ones) contain characteristic values of the objects. They form the description-vector for each conflict edge tested while constructing optimal schedule via Module 1. The first column enumerates conflict edges in the order as they have been tested in the B&B algorithm on the path to the optimal schedule in the solution tree (or to the best schedule obtained). The last column in table $T_{u,v}$ is used to classify objects in order to compare them with new objects within Module 3. At the solution stage (Module 3), the description-vector of the object is used to decide what edge substitution or removal has more probability to be successful.

Classes of Objects

To identify what transformation of conflict edge has been successful in the historical performance of the B&B algorithm, we introduce seven classes of objects: Ω_1, Ω_2,..., Ω_7 (see last column in Table 1). In what follows, it is assumed that inequality $i < j$ holds for each edge $[i, j] \in E$.

Object $[i, j]$ belongs to class Ω_1 if it is preferable to substitute this edge by arc (i,j) in order to obtain effective schedule with respect to the given objective function. (In such a case, we shall say that arc (i,j) *dominates* both arc (j, i) and edge $[i, j]$).

Object $[i, j]$ belongs to class Ω_2 if it is preferable to substitute it by arc (j, i), and we say that arc (j, i) dominates both arc (i,j) and edge $[i,j]$.

Object $[i, j]$ belongs to class Ω_3 if it is preferable to remove conflict edge $[i, j]$ from the mixed graph G, and we say that edge $[i, j]$ dominates both arcs (i,j) and (j, i).

If there is no enough information to decide what sole edge transformation is preferable, we have to include object $[i, j]$ to one of the classes Ω_4, Ω_5, Ω_6 or Ω_7 depending on available information. Namely, if arcs (i,j) and (j, i) simultaneously dominate edge $[i, j]$ but not each other, then $[i, j] \in \Omega_4$. If arc (i,j) and edge $[i, j]$ simultaneously dominate arc (j, i) but not each other, then $[i, j] \in \Omega_5$. If arc (j, i) and edge $[i, j]$ simultaneously dominate arc (i,j) but not each other, then $[i, j] \in \Omega_6$. Finally, if there is no enough information on arc (or edge) domination, then object $[i, j]$ has to belong to class Ω_7.

Next, we render the concrete edge classification rules realized within our Module 2. The learning process is based on the solution tree constructed for the sample problem. Let scheduling problem S be solved by an exact B&B algorithm within Module 1. Then we obtain optimal digraph $G[\alpha^*] = (V, A[\alpha^*], \varnothing)$ that defines optimal schedule. Moreover, the path from vertex G to vertex $G[\alpha^*]$ in the constructed solution tree is known. Each vertex of this path is defined by a conflict edge $[i,j]$. If $(i,j) \in A[\alpha^*]$, then $[i, j] \in \Omega_1$. If $(j, i) \in A[\alpha^*]$, then $[i, j] \in \Omega_2$. If $(i,j) \notin A[\alpha^*]$ and $(j, i) \notin A[\alpha^*]$, then $[i, j] \in \Omega_3$. Thus, each object obtained due to the solution of the sample problem via exact B&B algorithm belongs to one of the three classes: Ω_1, Ω_2, or Ω_3.

If the sample problem is solved only approximately, object classification is based on the relation between lower and upper bounds of the values of the given objective function. Let scheduling problem S be solved by an approximate *B&B* algorithm, then the best digraph $G[\alpha^\circ] = (V, A[\alpha^\circ], \varnothing)$ constructed gives upper bound UB that is greater than the minimal lower bound calculated for all the pendant vertices of the constructed solution tree. Thus, we obtain path $\pi(G_0 — G[\alpha^\circ])$ from vertex G_0 to vertex $G[\alpha^\circ]$ in the solution tree. Each intermediate vertex G' in this path defines conflict edge $[i, j]$ for further brunching. Object $[i, j]$ has to be included in one of the above seven classes. Next, we describe the rule for this classification in detail only for the case when path $\pi(G_0 — G[\alpha^\circ])$ includes arc (a) (see Fig. 1). We will indicate this case as follows: $(a) \in \pi(G_0 — G[\alpha^\circ])$. In this case, conflict edge $[i, j]$ in the mixed graph G is substituted by arc (i,j).

Let $(a) \in \pi(G_0 — G[\alpha^\circ])$. In the solution tree, three possible transformations have been realized for conflict edge $[i, j]$ (see Fig. 1). Let $LB(j, i)$ ($LB[i, j]$, respectively) denote the minimal lower bound calculated for all the pendant vertices G'' to which there exists a path $\pi(G' — G'')$ from vertex G' starting with arc (b): $(b) \in \pi(G' — G'')$ (starting with arc (c): $(c) \in \pi(G' — G'')$, respectively).

If $LB(j, i) \geq UB$ and $LB[i, j] \geq UB$, then $[i, j] \in \Omega_1$. If $LB(j, i) < UB$ and $LB[i, j] \geq UB$, then $[i, j] \in \Omega_4$. If $LB(j, i) \geq UB$ and $LB[i, j] < UB$, then $[i, j] \in \Omega_5$.

Classification rules for other two cases $(b) \in \pi(G_0 — G[\alpha^\circ])$ and $[c] \in \pi(G_0 — G[\alpha^\circ])$ are analogous. In particular, if $(b) \in \pi(G_0 — G[\alpha^\circ])$, then object $[i, j]$ has to belong to one of the three classes Ω_2, Ω_4 or Ω_6. If $(c) \in \pi(G_0 — G[\alpha^\circ])$, then object $[i, j]$ has to belong to one of the three classes Ω_3, Ω_5 or Ω_6.

At the end of the stage of learning, the information accumulated in table $T_{u,v}$ is used in Module 2 to construct the *recognition table,* which is used at the solution stage within Module 3. A recognition table has the similar form as table $T_{u,v}$ has, except the last row added to recognition table (see Table 1). (However, we use the same notation $T_{u,v}$ for recognition table as well.) In order to construct the recognition table a subset of the most informative characteristics is selected and sorted with respect to their informativeness. Some characteristics that turned out to be not essential for sample problems have to be rejected from table $T_{u,v}$.

In [13], it is shown how to select the most informative characteristics. Module 2 systematizes the knowledge obtained using the pattern recognition apparatus, and constructs the generalized logical rule for conflict resolution. Such a rule describes successful local decisions and should be used in Module 3 for more efficient solving large-scale scheduling problems via adopting successful local decisions by principle of analogy.

Solution Stage (Module 3)

Scheduling algorithm used as Module 3 occupies intermediate position between heuristic and approximate B&B algorithms depending on the number of sample problems solved exactly or approximately at the stage of learning (Module 1 and Module 2). Conflict resolution strategy is also used in Module 3. If Module 3 chooses conflict edge $[i, j]$, it has to recognize to which class from set $\{\Omega_1, \Omega_2,..., \Omega_7\}$ edge $[i, j]$ has to belong. Thus, a recognition problem arises, and it is reasonable to adopt to a new situation the strategy that has led to successes at the learning stage in Module 1.

In order to solve the recognition problem, Module 3 uses the procedure based on the bound calculation. A resemblance bound is calculated that characterizes the distance between the description vector of recognizable object $[i, j]$, and that of sample objects with respect to combination H of characteristics $x^1, x^2,..., x^u$ used in the recognition table $T_{u,v}$. We have to determine edges $[i_k, j_k]$ given in table $T_{u,v}$ such that the corresponding vectors $(b^1_k, b^2_k,..., b^u_k)$ are the closest ones to vector $(b^1, b^2,..., b^u)$ with respect to the characteristics $x^1, x^2,..., x^u$. Two objects $[i, j]$ and $[i_k, j_k]$ are considered to be similar with respect to the system H of characteristics, if at least δ inequalities $|b^j - b^j_k| \le \varepsilon_k$ are satisfied with $j = 1, 2,..., u$ and $x^j \in H$, where $\varepsilon_1, \varepsilon_2,..., \varepsilon_u$ and δ being parameters of Module 3. Bounds are calculated for the number of objects of the learning table that are similar (close) with respect to system H. The analysis of such bounds allows to decide to which class $\Omega_1, \Omega_2,..., \Omega_7$ object $[i, j]$ has to be assigned. Due to this calculation, one can decide what transformations are preferable for the conflict edge $[i, j]$, and how many transformations (one, two or three) of this edge have to be treated in Module 3. If $[i, j] \in \Omega_1$, then solution tree has to contain arc (a) (see Fig. 1). If $[i, j] \in \Omega_2$, then solution tree has to contain arc (b). If $[i, j] \in \Omega_3$, then solution tree has to contain arc (c). If $[i, j] \in \Omega_4$, then solution tree contains two arcs (a) and (b). If $[i, j] \in \Omega_5$, then solution tree contains two arcs (a) and (c). If $[i, j] \in \Omega_6$, then solution tree contains two arcs (b) and (c). If $[i, j] \in \Omega_7$, then solution tree contains all three arcs (a), (b) and (c) that are represented in Fig. 1.

Due to this approach, we obtain close interaction between implicit enumeration algorithm (Module 1), approximate algorithm and heuristics (in Module 3). Effective priority rules obtained via Module 2 contribute to the implementation of the effective upper and lower bounds in the branch-and-bound algorithm (Module 1). Suitable heuristics have to be incorporated in the branch-and-bound procedures (Module 1 and Module 3) to compute the solutions corresponding to the most promising part of the solution tree.

Conclusion

The use of an exact scheduling algorithm is computationally expensive and so impracticable. At present, rather common methodology carried out by practitioners is the use of simple heuristics based often on weakly grounded dispatching rules which do not ensure desirable accuracy of the obtained solution. To overcome these difficulties, we propose a knowledge-based system that may be used for the adaptation of the approximate version of the B&B algorithm for the specific practical scheduling problems. The solution process is partitioned into two stages: the *learning* stage realized by Module 1 and Module 2 for small (sample) scheduling problems, and the *solution* stage realized by Module 3 for moderate and large (real-world) scheduling problems.

Module 1 creates the database of learning information for the certain class of the scheduling problems and enlarges it while solutions of new problems become available. At the stage of learning, one can also use practically effective schedules obtained via commercial packages such as that described in [5]. The volume of information, its correctness and absence of noises depends on the quality of the scheduling algorithms applied in the framework of Module 1 and on the computing time given for learning process (Module 1 and Module 2). More exact algorithms requiring larger computing time may ensure more informative knowledge. Therefore, it is important to use effective algorithms within Module 1. The algorithm used in Module 3 should consider at each step the same question as the algorithm used in Module 1. The bridge linking Module 1 and Module 3 is

Module 2, which automates the learning of peculiarities of the similar scheduling problems and may give foundation for creating generalized logical rule for proper conflict resolution. As a result, Module 2 generates suitable priority rules, which may be considered in Module 3 as new comprehensive heuristics developed just for the practical scheduling problems under consideration.

This research was partially supported by INTAS (project 03-51-5501), and by the Belarusian Republican Foundation for Fundamental Research.

Bibliography

[1] Chan F.T.S., Chan H.K. Dynamic scheduling for a flexible manufacturing system – the pre-emptive approach. International Journal of Advanced Manufacturing Technology, 2001, 17 (10), 760–768.

[2] Dorndorf U., Pesch E. Evolution based learning in a job shop scheduling environment. Computers & Operations Research, 1991, 22 (1), 25–40.

[3] Hertz A., Widmer M. Guidelines for the use of meta-heuristics in combinatorial optimization. European Journal of Operational Research, 2003, 151, 247–252.

[4] Jain A.S., Meeran S. Deterministic job-shop scheduling: past, present and future. European Journal of Operational Research, 1999, 113, 390–434.

[5] Janicke W. Excel planning interface for the SAP R/3 system. Chemical Engineering & Technology, 2001, 24 (9), 903-908.

[6] Kim M.H., Kim Y-D. Simulation-based real-time scheduling in a flexible manufacturing system. Journal of Manufacturing Systems, 1994, 13 (2), 85–93.

[7] Muth J.F., Thompson G.L. Industrial scheduling. Englewood Cliffs, N.Y., Prentice Hall, USA. 1963.

[8] Nof S.Y., Grant F.H. Adaptive/predictive scheduling: review and a general framework. Production Planning and Control, 1991, 2 (4), 298–312.

[9] Panwalkar S.S., Iskander W. A survey of scheduling rules. Operations Research, 1977, 25, 45–61.

[10] Paternina-Arboleda C.D., Das T.K. A multi-agent reinforcement learning approach to obtaining dynamic control policies for stochastic lot scheduling problem. Simulation Modelling Practice and Theory (to appear, available online).

[11] Priore P., De La Fuente D., Gomez A., Puente J. A review of machine learning in dynamic scheduling of flexible manufacturing systems. AI EDAM, 2001, 15 (30), 251–263.

[12] Priore P., Fuente D.L., Gomez A., Puente J. Dynamic scheduling of manufacturing systems with machine learning. International Journal of Foundation of Computer Science, 2001, 12 (6), 751–762.

[13] Shakhlevich N.V., Sotskov Yu.N., Werner F. Adaptive scheduling algorithm based on mixed graph model. IEE PROCEEDINGS – Control Theory and Applications, 1996, 43 (1), 9–16.

[14] Sotskov Yu.N. Software for production scheduling based on the mixed (multi)graph approach. Computing & Control Engineering Journal, 1996, 7 (5), 240–246.

[15] Sotskov Yu.N. Mixed multigraph approach to scheduling jobs on machines of different types. Optimization, 1997, 42, 245–280.

[16] Tanaev V.S., Sotskov Yu.N., Strusevich V.A. Scheduling theory. Multi-stage systems. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994.

[17] Wang J.M., Usher J.M. Application of reinforcement learning for agent-based production scheduling. Engineering Applications of Artificial Intelligence, 2005, 18 (1), 73–82.

[18] Whiteson S., Stone P. Adaptive job routing and scheduling. Engineering Applications of Artificial Intelligence, 2004, 17 (7), 855–869.

[19] Wiers V.C.S. A review of the applicability of OR and AI scheduling techniques in practice. Omega – International Journal of Management Science, 1997, 25 (2), 145–153.

Authors' Information

Yuri N. Sotskov – United Institute of Informatics Problems of the National Academy of Sciences of Belarus, Surganova str., 6, Minsk, Belarus; e-mail: sotskov@newman.bas-net.by

Nadezhda Sotskova – Hochschule Magdeburg-Stendal, Fachbereich Wasserwirtschaft, PSF 3680, 39011 Magdeburg, Germany; e-mail: drsotskova@hotmail.com

Leonid V. Rudoi – United Institute of Informatics Problems of the National Academy of Sciences of Belarus, Surganova str., 6, Minsk, Belarus; e-mail: margol@newman.bas-net.by