Baltic J. Modern Computing, Vol. 2 (2014), No. 4, 272-284

Heuristic Reactive Rescheduling Algorithms in the Advanced Scheduling Add-on for ERP

Vytautas TIEŠIS[1], Gintautas DZEMYDA[1], Taskin KIZIL[2]

[1] Institute of Mathematics and Informatics, Vilnius University, LT-08663,Vilnius, Lithuania
[2] Hisbim IT Corp. Atap OSB, 26110 Eskisehir, Turkey

vytautas.tiesis@mii.vu.lt; gintautas.dzemyda@mii.vu.lt;
taskin@hisbim.com

Abstract. In the paper there are described some rescheduling algorithms which are implemented in the Advanced Planning and Scheduling system, namely PEN System, which is an add-on for the Enterprise Resource Planning system and intended to be used to mitigate the impacts of potential exceptions disrupting shop floor level schedules. The rather simple and quick heuristic algorithms, namely Affected Operations Rescheduling and Active Wilkerson-Irwin algorithm are chosen to cope with the machine disruption. The algorithms are modified for more complicated and realistic enterprise environment model than the Job Shop or the Flexible Job Shop, namely for complex operations' precedence constraints described by any simple directed acyclic graph.

Keywords: reactive rescheduling; heuristic algorithms; enterprise resource planning; precedence constraints.

1. Introduction

Advanced enterprise resource planning systems (ERP II) up to date stay ill-suited to deal with undesirable business events (machine break down, absence of employees, shortage of materials, etc.), other exceptions and with the optimisation of plans. That especially concerns ERP for small and middle enterprises. Such functionalities are often implemented as add-on modules that take data out of ERP databases and send the produced plans back for execution. The paper describes heuristic rescheduling algorithms which are implemented in the Advanced Planning and Scheduling (APS) system, namely PEN System (Caplinskas et al., 2012), which is an add-on for the ERP system and intended to be used to mitigate the impacts of potential exceptions disrupting shop floor level schedules an mid-term level production plans. The PEN System is prepared to proceed with variety of rescheduling algorithms; and system architects should choose most proper algorithms in the process of design specific architectures in the particular target enterprises on the basis of PEN System. In the paper quick heuristic algorithms able to cope with the machine disruption are presented − Affected Operations Rescheduling (AOR) (Abumaizar and Svestka, 1997) and Active Wilkerson-Irwin algorithm (AWI) (Dong,Y.-H. and Jang, J., 2012; Wilkerson, L.J. and Irwin, J.D., 1971). The first algorithm shifts to right exclusively affected operations on the same machines preserving processing order. The second algorithm dispatches operations on machines in the manner that creates active schedule for the Job Shop heuristically seeking improve

tardiness, make-span and stability (minimal deviation of new operations' start times from the original start times). The AWI with straightforward modification may be applied to Flexible Job Shop where affected operations may be dispatched to alternative machines. The modifications of algorithms for more complicated and realistic enterprise environment than Job Shop and Flexible Job Shop are presented in the paper. The model of production includes precedence constraints that may be described by any simple directed acyclic graph: each operation may have several precedents and several successors. The model may be regarded as a simultaneous execution of several independent projects.

The elaborated models of the enterprise environment and the ones of schedules are described in the Section 2. The modifications of heuristic algorithms are presented in the Section 3. The complexity of algorithms and their application is discussed in the Section 4.

2. Models

Concepts of the enterprise environment and schedules are described in this section, namely the denominations of parameters of resources, production and schedule and there relations are described.

2.1. Resources

The algorithms presented in the paper deal with one kind of technical resource – the machines. In general, the machine is a device which is able to perform one or several technological operations. The concepts and denominations related to resources are described below.

M is a set of machines. The total number of machines is denominated by $m = |M|$.

r is a number (identifier) of a machine.

An operation is defined as a basic technological operation that is not decomposed in simpler technological operations in a schedule. To meet the requirements of presented algorithms, an operation is described by:

- the set of machines that are able to perform it;
- processing time (setup time is included in the processing time).

An operation is denominated as an ordered pair $(j,k)\epsilon O_j$ where j is a number of a job, k is a number of an operation in the j–th job operations' list $O_j = (j,1); \ldots; (j,k); \ldots; (j,o_j)$, where the j–th job has o_j operations.

p_{jk} is processing time of the operation $(j,k)\epsilon O_j$. It is fixed, same for all alternative machines and known a priori.

$M_{jk} \subset M$ is a set of machines which are able to perform operation $(j,k)\epsilon O_j$.

2.2. Production and jobs

For the production on the shop floor the presented algorithms use the model which includes precedence constraints of the job's operations that may be described by any simple directed acyclic graph. A number of jobs are processed on a number of machines. The processing of a job on a machine is called an operation. The processing time p_{jk} of each job on each machine is known and fixed. Each machine can process only one job at

a time and preemption is not allowed. For each operation alternative machines may exist. Different jobs may visit different machines. The graph of operations' precedence $G(O_j)$ (i.e., the job routing) is individual for each job and is known a priori, an operation may have several immediate successive operations and several directly predeceasing operations. The time of release and the due time should be defined for a job. There are no relations of precedence among operations of different jobs. Below the denominations related to the production are described.

J is a set of jobs. The total number of jobs is denominated by $n = |J|$.

r_j is a release date of a job $j \epsilon J$.

d_j is a due date of a job $j \epsilon J$.

$G(O_j)$ is the simple directed acyclic graph which correspond to the direct precedence constraints of the operations $(j,k) \epsilon O_j$ of job j. The operations are nodes, and arcs represent precedence relations.

succ(j,k) is a set of the immediate successive operations of an operation (j,k) in the graph $G(O_j)$. In the case when directed graph is intree (any operation has at most one successor) the succ(j,k) consist of one operation or is an empty set for the last operation.

pred(j,k) is a set of directly predeceasing operations of an operation (j,k) in $G(O_j)$. In the case when directed graph is outtree (any operation has at most one predecessor) the pred(j,k) consist of one operation or is an empty set for the first operation.

If operations' graph $G(O_j)$ constitute a simple sequence of operations (succ(j,k) and pred(j,k), both consist of one operation or both are empty sets) then $O_j = (j,1); \dots; (j,k); \dots; (j,o_j)$ is a chain of operations ordered according to this sequence. The Job Shop Scheduling problem uses this model and many academic algorithms are proposed for this problem (Pinedo, 2012). In all cases the operation (j,o_j) is the last, predeceased indirectly or directly by other operations from $G(O_j)$. If the original operations' graph has several terminal nodes then (j,o_j) is a mock node with a processing time equal to zero.

2.3. Undesirable Business Events

The Undesirable Business Event is unexpected facts or circumstances which disturb the normal implementation of the actual running schedule. Many Undesirable Business Events can be modelled as machine breakdowns since they involve a disruption in the processing of operations on a machine or machines for a period of time. The event is described by a time stamp, an impact estimation (in time), related processes and resources:

Dm is the broken machine, $Dm \epsilon$M.

Rt is the time when the broken machine is anew available for processing.

Ew is the end of Time Window (Rescheduling Point) for the rescheduling. The rescheduling is invoked only by events with significant impact therefore $Ew \ll Rt$.

Evs is the time stamp when the disruption has been started.

(jo, io) is the interrupted operation.

2.4. Schedule

Rescheduling algorithms reschedule the Actual Running Schedule which is the schedule that has been running until the Undesirable Business Event has occurred and the Event

has been decided by the decision maker or accepted automatically to be a reason for the rescheduling. Such concepts describe models of schedules:

s_{jk} is the planned start time of an operation (j,k) in the Actual Running Schedule;

c_{jk} is the planed completion time of an operation (j,k) in the Actual Running Schedule.

$m_{jk} \in M_{jk}$ is the machine on which the operation (j,k) is assigned to process.

JO_r is the job order for the machine $r \in [1,m]$, the processing sequence of operations on the machine r. The job orders and operation start times are data forming the Gantt chart. The order $JO_r = (op_{r1}, \dots, op_{rl}, \dots, op_{r,lm_r})$ is unambiguously determined by the sequence of start times $s_{jk}, j \epsilon J, k = 1, o_j$, $(\forall\ j,k : m_{jk} = r)$. Therefore for each $l \in 1, lm_r$ there exist the relation between operations' list JO_r for r machine and list O_j for j job: $op_{rl} \equiv j,k$, where the operation j,k is assigned to the machine $m_{jk} = r$.

The operations in the Actual Running Schedule should be separated out into operations to reschedule and operations which will be successfully started until the end of the calculation Time Window. It is supposed that start times of operations determined in the Actual Running Schedule are kept at the processing on the shop floor during the calculation Time Window. Bellow the process of the separation is described.

JR is the set of remaining to reschedule job numbers. Initially $JR = J$; the job j is excluded from JR if all operations from $O_j = (j,1); \dots; (j,k); \dots; (j,o_j)$ are marked as finished, i.e. if the last operation (j,o_j) is marked as finished.

$Fin0$ is the set of numbers (j,k) $j \epsilon J$ of operations that have been finished at the time stamp *Evs* of rescheduling calculations. If the final operation (j,o_j) from O_j is included in $Fin0$ then the job j is excluded from JR.

$DirAff$ is the set of numbers (j,k) of operations which are affected by the undesirable business event. If the unexpected event is "broken machine" then affected operations are those:

1) Operations included in the broken machine Dm job order $JO(Dm)$ wich planed completion time c_{jk} is after disruption start time Evs or the planned start time s_{jk} is after Evs and before the time Rt when the broken machine is available;
2) Operations that are in directed graphs $G(O_j)$ after operations settled in 1); those operations can be determined by Breadth-First search (Knuth, 1997) for each $j \epsilon J$.

$Fin1$ is the set of numbers (j,k) of operations that will be started until the end of the rescheduling Time Window (the planed start time s_{jk} is less than Ew) and which are not included in the $DirAff$. If all o_j operations from O_j are included in $Fin0$ and $Fin1$, namely if the final operation (j,o_j) is included in $Fin1$, then the job j is excluded from JR

Rem_{jk} is the indicator of remaining operation (*j*,*k*) to be rescheduled, $Rem_{jk} \in \{0,1\}$, $j \in JR, k = 1, o_j$ (all operations excluding sets of finished operations *Fin*0 and *Fin*1 have indicator $Rem_{jk} = 1$)

On purpose to simplify the rescheduling it is reasonable to let only remaining operations in the plan (in the lists GO and JO).

$Plan1$ – new rescheduled schedule with new start times $s1_{jk}$ of operations.

3. Rescheduling algorithms

Usually there is a short Time Window for rescheduling calculations and corresponding decision making. Therefore the rough methods are often used for rescheduling that produce rather effective but not necessarily optimal plans. Several heuristic approaches are chosen for rescheduling in the PEN system.

Right-shift rescheduling is one of methods for partial rescheduling (Abumaizar and Svestka, 1997). It is simple, quick to calculate, but generates a schedule far from optimal – it shifts all remaining operations by the duration of the damaged machine reparation. It is applicable only in the case of redundant resources, flexible schedules or when customer orders may wait. Right-shift rescheduling algorithm serves as the base benchmark for evaluation of elaborated algorithms. The method completely preserves the initially scheduled operation's sequences including operations on the damaged machine that are processed after the reparation of the machine.

More effective method applies right-shift for only those operations that are affected directly or indirectly by the disruption; the method is called the Affected Operations Rescheduling (AOR) (Abumaizar and Svestka, 1997). The aim of the method is to preserve the initial schedule as much as possible, in order to minimize the disruptions of the raw materials' and parts' delivery schedule. The algorithm does not search for alternative machines; operations on the damaged machine are processed after the reparation of the machine, like in the Right-shift case.

Differently from the Affected Operations Rescheduling, the Active Wilkerson Irwin algorithm (AWI) (Dong,Y.-H. and Jang, J., 2012; Wilkerson, L.J. and Irwin, J.D., 1971) with minor modification may allocate the operations that have been interrupted because of the machine damage on the alternative operable machines. The heuristic method of quick rescheduling tries to optimize schedules allocating operations in course by some heuristic rules. The allocation rules also seek to minimize deviation from the actually running schedule. In many cases the AWI overcomes AOR, in some cases does not, especially in the aspect of stability. Therefore the quick AOR serves in the PEN system as a generator of an optional rescheduled plan to choose by the decision maker.

3.1. Affected Operations Rescheduling algorithm

An algorithm for rescheduling the affected operations in a job shop is presented in the paper (Abumaizar and Svestka, 1997). The results of experiments demonstrate that the Affected Operations Rescheduling (AOR) avoids the disadvantages associated with the full-scale rescheduling and Right-Shift rescheduling methods. In this section the modification of the AOR is presented when the model of production includes precedence constraints of the job's operations that may be described by any simple directed acyclic graph $G(O_j)$. The model may be regarded as a mutual execution of several independent projects. Furthermore, our algorithm takes into account a fact that non-affected operations are processed during the rescheduling Time Window.

The basic principle of AOR is to react to any disruption by delaying affected operations' starting times (pushing them forward) by the minimum amount required to:

- Keep the technologically constrained precedence relations of job's operations;
- Preserve the initially scheduled sequence of operations on each machine. In the original AOR (Abumaizar and Svestka, 1997) the initial sequence of

operations is strictly preserved because it is supposed that the production process is stopped during the rescheduling Time Window.

In our modification it is supposed that the unaffected operations are processed during the Time Window. The starting times of such operations are kept as in the Actual Running Schedule. So there are some permutations at the beginning of sequences of operations remaining to reschedule (awaiting of rescheduling).

Let us describe the idea of AOR in the Job Shop case. The technologically constrained and scheduled sequences of job's and machine's activities may be represented as a binary tree (see Fig. 1). The nodes represent the operations (j,k). Two branches exit the node (j,k). One branch, namely the machine branch, points to the operation next on the machine (NOM) according to the operations assignments in the Actual Running Schedule, i. e. according to the job order JO_r where $r = m_{jk}$. Another one, namely job branch, points to the operation next of the job (NOJ) immediate successive operation succ(j,k) according to the technological precedence constraints described by the graph $G(O_j)$.

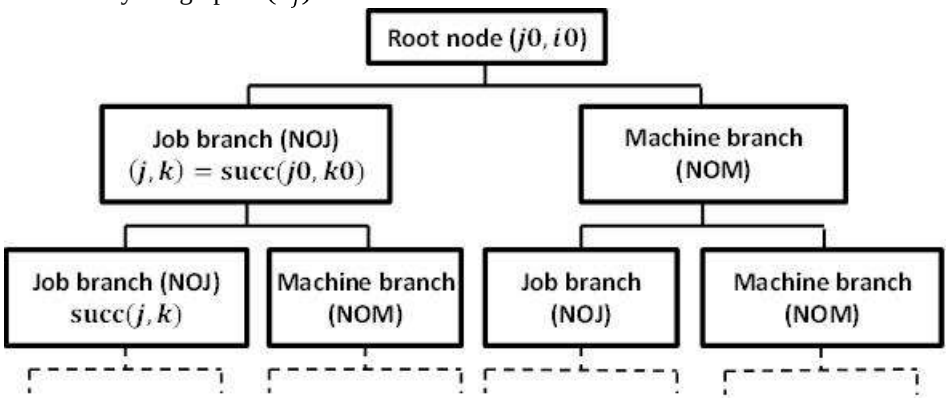

Fig. 1 The binary tree of the delay propagation

Algorithm starts with the root node, that represents the starting affected operation (jo, io), often (jo, io) is an interrupted operation and propagates delay through the binary tree identifying other affected operations. To find out if the operation (j,k) is affected, its planned start time s_{jk} is compared with the shifted completion time of the preceding operation ((jo, io) in the job branch in Fig. 1). If the completion time of the preceding operation is less than the s_{jk}, then the operation (j,k) is assumed to be unaffected and no branching from (j,k) is needed. Furthermore, no more shifts are performed on this branch. If the completion time of the current operation is greater than s_{jk}, then the operation (j,k) is affected and the following additional calculations need to be performed:

- update s_{jk} – make it equal to the completion time of the preceding operation and update the completion time of (j,k);
- include the operation (j,k) into the set of the affected operations for further branching.

In the case of several successive operations succ(j,k) there may be several job branches. In the case of several preceding operations pred(j,k) the operation (j,k) may appear in several job branches and as a consequence it may be shifted many times prolonging a run time of rescheduling.

The algorithm uses such internal variables:

$c1_{jk}$ is a completion time of an operation (j,k) according to the rescheduled schedule *Plan1*.

cp_{jk} is a temporal completion time of the potentially affected operation.

sp_{jk} is a temporal start time of the potentially affected operation.

Q is an object – the queue of potentially affected operations (j,k).

Q.push(*j*,*k*) is a method: push (j,k) into the queue, the simplest method is to include it at the end of the queue.

Q.pop is a method: take the first pair from the queue.

$Q.q_{jk}$ is a property, the indicator of an inclusion of the operation into the queue. $Q.q_{jk} \in \{0,1\}$, $Q.q_{jk} = 1$ if (j,k) is included into Q.

Aff_{jk} is the indicator of an operation affected by the right shift; $Aff_{jk} \in \{0,1\}$, $Aff_{jk} = 1$ in the affected case.

Makespan is a makespan of the rescheduled schedule (the completion time of the last job).

$devSt$ is a total starting time deviation between the initial schedule and the new one: $devSt = \sum_{jk} s1_{jk} - s_{jk}$.

$F(\mu)$ is the earliest free time on the machine μ.

Below the pseudo-code of the algorithm with comments is presented.

Step 1. Choose the starting potentially affected operation $(j1,k1)$ as one of the following:

(a) The interrupted operation (jo,io) if it exists;

(b) If there is no interrupted operations, then as $(j1,k1)$ choose from remaining operations on the damaged machine Dm if such an operation exists and if its start time $s_{j1,k1}$ is less than the ready time Rt.

(c) Otherwise, the algorithm terminates and there are no affected operations (i.e. no need for rescheduling).

Take the starting potentially affected operation as a current operation: $(jc,kc) = (j1,k1)$.

Step 2.

Makespan = 0; *devSt* = 0.

The earliest free times $F(\mu)$ on the machines $\mu \in M$ are calculated. The $F(\mu)$ is equal to the maximum selected from the End of Time Window Ew and the completion time of the last unaffected operation that may be started on machine μ until Ew:

$F\ Dm = Rt$; // the time when the broken machine Dm is available for processing.

For $\mu \in M, \mu \neq Dm$ do

$l = 1$, $cmax = 0$;

$j,k = op_{\mu,l}$ from $JO_{mc} = op_{\mu,1}, \dots, op_{\mu,l}, \dots, op_{\mu,lm_\mu}$;

While $s_{jk} < Ew$ and $l \leq lm_\mu$ do

$cmax = c_{jk}$;

l=l+1;

If $l \leq lm_\mu$ then $j,k = op_{\mu,l}$;

End of While.

$F\ \mu = \max\ Ew, cmax$.

End of For.

Set initial data for all remaining operations:

For all $j,k : (j \in JR, k\epsilon\ 1..o_j\ )$ assign
$Aff_{jk} = 0$; // j,k is not affected
$Q.q_{jk} = 0$;
$s1_{jk} = s_{jk}$;
sp_{jk}= max{$F\ m_{jk}\ , s1_{jk,}\ Ew$};
$cp_{jk} = sp_{jk} + p_{jk}$;
If $s1_{jk} \neq sp_{jk}$ then $Q.push(v)$; $Q.q_{jk} = 1$; End of If.
End of For.
//----------Below is the main loop----------

Step 3. If the current operation (jc, kc) has not been marked as an affected operation mark it as an affected operation*:*
If $Aff_{jc,kc} = 0$ then $Aff_{jc,kc} = 1$;

Step 4. Update the schedule, id est delay the current operation according to its temporal start time $sp_{jc,kc}$ and update schedule's quality indexes:
$devSt\ = devSt + max\quad sp_{jc,kc} - s1_{jc,kc}\ , 0\ ;$
$s1_{jc,kc} = sp_{jc,kc}$;
$c1_{jc,kc} = cp_{jc,kc}$;
$Makespan = max\quad Makespan,\ c1_{jc,kc}\ .$

Step 5 . Operations next to the current operation (in the Step 5 the next ones in the job's technological graph; in the Step 6 the next ones on the same machine $m_{jc,kc}$) are included into the queue Q of potentially affected operations if the new completion time $c1_{jc,kc}$ of the current operation impacts the start time of the next operation:
If $succ\ jc, kc\ = \emptyset$ then go to step 6;
For all $v\epsilon succ\ jc, kc$ do //For all successive operations
If $sp_v < c1_{jc,kc}$ then
$sp_v = c1_{jc,kc}$
$cp_v = sp_v + p_v$;
If $Q.q_v = 0$ then
$Q.push(v)$;
$Q.q_v = 1$;
End of If;
End of If
End of For.

Step 6. Operation next to the current operation on the same machine is included into the queue of potentially affected operations:
$mc = m_{jc,kc}$; //Machine (workplace) of the current operation
$l = num_{jc,kc} + 1$; //serial number of the next operation in the jobs order on the machine mc
If $l \leq lm_{mc}$ then
$v = op_{mc,l}\ from\ JO_{mc} = \ op_{mc,1}, \dots, op_{mc,l}, \dots, op_{mc,lm_{mc}}\ ;$
If $sp_v < c1_{jc,kc}$ then
$sp_v = c1_{jc,kc}$;
$cp_v = sp_v + p_v$;
If $Q.q_v = 0$ then
$Q.push(v)$;
$Q.q_v = 1$;

End of If;
End of If;
End of If.

Step 7. If the queue of potentially affected operations is empty, then stop, otherwise take an operation from the queue and return to step 3:

If $Q = \emptyset$ then STOP

Else *(jc,kc)=Q.pop,* $Q.q_{jckc} = 0$; Goto Step 3.

3.2. Modified Wilkerson Irwin algorithm

The algorithm is based on the algorithms proposed in (Dong and Jang, 2012; Wilkerson, L.J. and Irwin, J.D., 1971) for a classical Job Shop scheduling problem. We have adapted the algorithm to the PEN System model (see section 2). The modifications consist of these enhancements:

- The Flexible Job Shop model is considered instead of Job Shop; the operations are processed in work centres with alternative machines. So affected operations may be dispatched to alternative operable machines instead of waiting until the damaged machine is repaired. This generalisation also changes procedures of earliest start time calculation because the start time is looked up through alternative machines. The operations, candidates to reschedule, are also looked up through the whole work centre on which the earliest completion time can be achieved (see Step 4 below).
- The used model of production is even more complex than Flexible Job Shop – it includes precedence constraints of the job's operations that may be described by any simple directed acyclic graph. This generalisation of the production model also increases amount of calculations.

The algorithm has two heuristic variants of rescheduling – MWI-J (Modified Wilkerson Irwin–Job) and MWI-O (Modified Wilkerson Irwin–Operation). Both variants are based on the Wilkerson–Irwin algorithm that is well known for minimising average tardiness of a schedule. MWI-J tries to minimise heuristically the efficiency measures such as mean tardiness, mean flow time and mean makespan with less concern about stability. On the other hand, MWI-O attempts to balance efficiency and stability (pursuance of original operations' start times in the rescheduled plan).

At each stage of the algorithm an operation is rescheduled. So, in the pseudo-code there are used some additional variables that are indexed by the stage number t:

$Plan1(t)$ is the partial schedule containing t rescheduled operations;

S_t is the set of schedulable operations at the stage t corresponding to given $Plan1(t)$. A schedulable operation is either the initial operation of a job remaining to reschedule or an operation whose direct predecessors are already scheduled in the $Plan1(t)$;

se_{jkt} is the earliest time when operation $(j,k) \in S_t$ can start;

do_{jk} is the provisional due date of the operation (j,k) determined at Step 1 of the procedures MWI-J and MWI-O

In each iteration the new operation is added to the partial schedule $Plan1(t)$ containing t scheduled operations. Initially the $Plan1(0)$ is empty.

At first the procedure of schedule generation determines a target machine m^*, on which we can get the earliest completion time c^* among the completion times of all schedulable operations.

Then, the procedure selects one of schedulable operations on the target machine whose earliest start time se_{jkt} is earlier than the earliest completion time c^*, and makes new partial schedule $Plan1(t+1)$ by adding the selected operation to the current partial schedule $Plan1(t)$. The new schedule properties depend on the operation's selection rule. Two rules are elaborated. The rule in MWI-J tries to minimise the efficiency measures such as mean tardiness, mean flow time and mean makespan. The rule in MWI-O tries to balance efficiency and stability defined as deviation of new operation start times from the original start times

After failure of a machine Dm, the interrupted operation (jo, io), all subsequent operations in the Jobs Order $JO(Dm)$ for the interrupted machine and all operations that cannot be successfully started until the end of the Time Window are subject to rescheduling. All those operations j,k are marked by $Rem_{jk}=1$ (see section 2.4). The procedure of MWI-J is as follows:

Step 1: If in the Actual Running Schedule there are no scheduled operations on the damaged machine *Dm* during an interval of reparation [*Evs*, *Rt*] then there is nothing to reschedule.

The auxiliary Right Shift is proceeded to calculate provisional due dates do_{jk} of the operations $j,k : Rem_{jk}=1$. The auxiliary starting and completion times of operations $(sr_{j,k}, cr_{j,k})$ are calculated as described below.

The last operations $(j, o_j) \in O_j$ of all jobs $j \in JR$ are shifted forward in time until they meet their job due dates d_j: $cr_{j,o_j} = d_j$. If previously a solution of lateness was accepted and a job is already late in the Actual Running Schedule, the last operation of the job is kept at its current place.

All other operations are sequentially shifted to the right starting from the one having a latest completion time c_{jk} in the Actual Running Schedule. The operation $j,k : Rem_{jk}=1$ is being shifted until its completion time $cr_{j,k}$ meets the minimal start time $sr_{succ(j,k)}$ from start times of all successive operations $succ(j,k)$ or the start time of the successive operation $sr_{\boldsymbol{op}_{m_{jk},\boldsymbol{num_{jk}+1}}}$ on the machine m_{jk} in the Job Order $JO_{m_{jk}} = op_{m_{jk},1}, \dots, op_{m_{jk},l}, \dots, op_{m_{jk},lm_{m_{jk}}}$:

$cr_{j,k} = \min \min_{(j,k1)\in succ(j,k)} sr_{j,k1}, sr_{op_{m_{jk},num_{jk}+1}}$.

This right shift of operations keeps the original sequence of operations on each machine and feasibility of the scheduling problem (regardless of the failure of the machine Dm). Let those shifted completion times $cr_{j,k}$ be the provisional due dates do_{jk} of the operations.

Step 2: (Initialisation) Let t=0, $Plan1\ t = \emptyset$.

Let S_t be the set of all schedulable operations corresponding to the $Plan1\ t$, i.e. for each job $j \in JR$ a schedulable operation $j,k \in S_t$ is an operation with $Rem_{jk}=1$ and is either the initial operation of the job (pred $j,k = \emptyset$) or its direct predecessors $\text{pred}(j,k)$ should have started before Ew – the End of the Time Window:

For each $j \in JR$, $k\epsilon(1..o_j)$ do

If (pred $j,k = \emptyset$) then j,k include into S_t

The earliest available time $F(Dm)$ for a next operation on the broken machine Dm is equated to the repair completion time of the broken machine Rt: $F\ Dm = Rt$. Obviously, the Rt is greater than the End of Time Window Ew (Rescheduling Point).

For all other machines $\mu \in M$, the $F(\mu)$ equals to the completion time c_{jk}: $\mu = m_{jk}$ of the last operation which can be processed on the machine μ at the Time Window or equals to the Ew, depending which time is later. The calculation of $F(\mu)$ is the same as in Step 2 in Section 3.1.

Calculate the earliest times se_{jkt} at which operations $j,k : (j,k) \in S_t$ can start and find corresponding machine with earliest feasible time:

$$se_{jkt} = \max\{\min_{m_{jk} \in M_{jk}} F(m_{jk}), \max_{(j\_,k\_) \in pred\ jk} c_{(j\_,k\_)}\}$$

$$m_{jkt} = \arg\min_{m_{jk} \in M_{jk}} F(m_{jk})$$

Step 3: (Earliest completion time) Determine earliest completion time $c^* = c_{j^*k^*} = \min_{(j,k) \in S_t}(se_{jkt} + p_{jk})$ and the machine $m^* = m_{j^*k^*t}$ on which completion time c^* can be realised.

Step 4: (Candidate operations) Select all operations $(j,k) \in S_t$ whose earliest start time is less than c^* and that require a machine from the same work centre as the machine m^***:** select $j,k : (m_{jk} \in M_{j^*k^*}$ and $se_{jkt} < c^*)$.

If only one operation is selected, let it be operation (j^*k^*) selected to reschedule and go to Step 5. If there are more selected operations, use the following rules to select for rescheduling the next operation on the work centre with the machine m^*:

Rule 1. Let operations $(j1,k1)$ and $(j2,k2)$ be the operations whose job due dates d_{j1} and d_{j2} are among the selected operations the smallest and second smallest, respectively.

Rule 2. If $(\max\ se_{j1,k1,t} + p_{j1,k1}, se_{j2,k2,t} + p_{j2,k2} \le do_{j2,k2})$ or if $(se_{j1,k1,t} + p_{j1,k1} \le se_{j2,k2,t} + p_{j2,k2})$ then the operation $(j^*k^*) = (j1,k1)$ is selected to reschedule, else the operation $(j^*k^*) = (j2,k2)$ is selected to reschedule. Define $m^* = m_{j^*k^*t}$, where $m_{j^*k^*t}$ is the machine with earliest start time $se_{j^*k^*t}$ for operation (j^*k^*).

Step 5: (Update) Set the earliest available time of machine m^* equal to the completion time of operation (j^*k^*), i.e. $F\ m^* = se_{j^*k^*,t} + p_{j^*k^*}$. Set the completion time of the operation (j^*k^*), i.e. $c_{j^*k^*} = se_{j^*k^*,t} + p_{j^*k^*}$.

Include the operation (j^*k^*) into $Plan1(t)$ with start time $se_{j^*k^*,t}$ on the machine $m^* = m_{j^*k^*t}$ creating in such way the $Plan1(t+1)$. Remove operation (j^*k^*) from S_t. Increase t by 1.

Consider all successors of the operation (j^*k^*). If the operation $(j,k) \in succ(j^*k^*)$ that is the direct successor of operation (j^*k^*) may be already proceeded, i.e. if all $pred\ j,k$ are already scheduled then form S_t by adding (j,k) to S_{t-1}.

Calculate the earliest starting time of the operation (j,k):

$$se_{j,k,t} = \max\ \{\min_{m_{j,k} \in M_{j,k}} F(m_{j,k}), \max_{(j,k) \in pred\ (j,k)} c_{(j,k)}\} \qquad (1)$$

By the use of the same formula (1) recalculate earliest starting times of other schedulable operations $\{j,k : j,k \in S_t, m^* \in M_{j,k}\}$ which may be processed on the machine $m^* \in M_{j,k}$ because it's earliest available time $F\ m^*$ have been changed.

If S_t is empty, stop the procedure; otherwise, go to Step 3.

End of Algorithm.

The MWI-O algorithm is the same as the MWI-J algorithm, except the Rule 1 in the Step 4; while MWI-J uses job due dates d_j, MWI-O uses an operation due dates do_{jk} to increase stability of the rescheduling:

Rule 1. Let operations $(j1, k1)$ and $(j2, k2)$ be the operations whose due dates do_{j1k1} and do_{j2k2} are among the selected operations the smallest one and second smallest one, respectively.

4. Discussion and Conclusions

Two quick heuristic algorithms are chosen and modified for the PEN system. They are recommended to implement in the particular target enterprise Advanced Planning and Scheduling (APS) system on the basis of PEN System in the case of frequent disruptions, short rescheduling Time Window and limited calculation resources.

Let us consider the Job Shop case. In this case the operations' graph $G(O_j)$ is a chain of remaining operations of j job and the Affected Operations Rescheduling (AOR) algorithm is very quick. Let R be the number of remaining operations and $N \leq R$ be the number of affected operations. All calculations of initial two steps may be proceeded looping through all remaining operations therefore the complexity of those steps is $\boldsymbol{O}\ R$. In the Job Shop case the main loop has no inner loops and each affected operation may be shifted twice – once in the job branch and once more in the machine branch. Therefore the complexity of the AOR algorithm is $\boldsymbol{O}\ R + 2N$. All calculations of initial two steps of the AWI algorithm also have complexity $\boldsymbol{O}\ R$. One of remaining operations is rescheduled in the main loop and the number of remaining operations decreases by one. Therefore the complexity of the main loop (steps 3-5) is $\boldsymbol{O}\ \ R-1\ R\ _2$ and the complexity of the AWI algorithm is $\boldsymbol{O}\ R^2$, that is greater than of AOR algorithm, particularly when the number of affected operations is small.

The imperfection of AOR is that interrupted operations which were initially scheduled on the damaged machine should wait until the damaged machine will be fixed. AOR works quickly and produces good plans only when the initial plan is flexible and reparation time of the damaged machine is short.

When there are alternative machines processing the same operation, then the Wilkerson Irwin algorithm may be used to allocate on the alternative operable machine the operations which have been interrupted because of the damaged machine. Let us consider the case when the precedence graph $G(O_j)$ is any directed graph. All calculations of initial two steps of the AWI algorithm have complexity $\boldsymbol{O}\ (R+m)R$ because inner lops scan through predecessors, successors and machines. The Steps 3, 4 and 5 of the algorithm require $\boldsymbol{O}\ (R+m)R$ calculations in the worst case, therefore the main loop requires $\boldsymbol{O}\ (R+m)R^2$ calculations. Consequently, the complexity of the AWI algorithm still is polynomial - it requires $\boldsymbol{O}\ (R+m)R^2$ calculations.

Acknowledgements

The research has been supported by the EUROSTARS Project E!6232-PEN "Production effectiveness navigator". The authors cordially thank all the project partners for contribution and the Agency for Science, Innovation and Technology (Lithuania) for the financial support of the research.

References

Abumaizar, R.J., Svestka, J.A. (1997). Rescheduling job shops under random disruptions. *Int. J. Prod. Res.* 35, 2065–2082.
Caplinskas, A., Dzemyda, G., Kiss, F., Lupeikiene, A. (2012). Processing of Undesirable Business Events in Advanced Production Planning Systems. *Informatica*. 23, 563–579.
Dong,Y.-H., Jang, J. (2012). Production rescheduling for machine breakdown at a job shop. *Int. J. Prod. Res.* 50, 2681–2691.
Giffler, B., Thompson, G.L. (1960). Algorithms for solving production scheduling problems. *Oper. Res.* 8, 487–503.
Knuth, D. E. (1997). *The Art of Computer Programming*. Vol 1. 3rd ed., Addison-Wesley, Boston.
Pinedo, M.L. (2012). *Scheduling. Theory, Algorithms, and Systems*. 4th ed., Springer, New York.
Wilkerson, L.J., Irwin, J.D. (1971). An improved algorithm for scheduling independent tasks. *AIIE T.* 3, 239–245.

Authors' information

Vytautas Tiešis, is the research fellow at the Vilnius University Institute of Mathematics and Informatics. His main research interests include optimisation algorithms, scheduling, knowledge discovery and forecasting, and statistical analysis.

Gintautas Dzemyda, habil. dr., is a member of Lithuanian Academy of Sciences, professor, principal researcher and director of the Vilnius University Institute of Informatics and Mathematics. His main research interests include optimization and visualization, data mining, and medical informatics.

Taskin Kizil, is General Manager of Hisbim IT Corporation. His interests include high quality software solutions in IT Sector for the Enterprise and Manufacturing, Cognitive Systems and Robotics.

Received January 9, 2014, revised February 20, 2014, accepted February 27, 2014