Stability-Oriented Evaluation of Hybrid Rescheduling Methods in a Job-Shop with Machine Breakdowns

András Pfeiffer[1], Botond Kádár[1], László Monostori[1,2]
[1]Computer and Automation Research Institute Hungarian Academy of Sciences
Kende u. 13-17, Budapest, H-1111, Hungary
[2]Department of Production Informatics, Management and Control, Faculty of Mechanical Engineering
Budapest University of Technology and Economics, Budapest, Hungary
E-mail: pfeiffer@sztaki.hu

Abstract
The paper gives a review of the literature related to the stability measures and describes a new concept with exhaustive simulation results. Predictive production schedules are calculated by a scheduler based on a genetic algorithm, hybridized with a modified Giffler&Thompson algorithm. By applying the proposed scheduler and executing the resulted schedules by simulation, the solution methods for stability-oriented rescheduling described above can be profoundly tested and analyzed. Evaluation of several scenarios of the rescheduling threshold and the timing of rescheduling are presented in a case study. The system to be (re)scheduled is a five-machine test system with machine breakdowns. The results show how the investigated rescheduling parameters and the disturbances generated into the system affect the efficiency and the stability of the schedule execution.

Keywords
Dynamic Scheduling, Rescheduling, Simulation, Stability, Machine Breakdown

1 INTRODUCTION

In manufacturing systems, difficulties arise from unexpected tasks and events, non-linearities, and a multitude of interactions while attempting to control various activities in dynamic shop-floors. Complexity and uncertainty seriously limit the effectiveness of conventional control and (predictive) scheduling approaches. The selection of the most appropriate scheduling/rescheduling algorithms or the timing of the rescheduling action for a given assignment is not a trivial task, which, however, can be supported by simulation-based evaluation.
Previous studies in the literature mostly consider only two main goals defined for the rescheduling action:

- make the schedule executable/feasible again,
- improve the efficiency performance measure due to adaptation of the schedule to the situation occurred.

In the recent years, as the third goal, several studies deal with the effect of the rescheduling also from the stability point of view. The aim of the paper is to analyze the control action taken by the scheduler on several rescheduling scenarios especially focusing on the timing in a small job-shop environment. We present a stability measure and a stability-oriented schedule calculation method – based on a genetic algorithm – to be able to minimize the negative effect of the changes induced by the rescheduling, however, keeping efficiency also at considerable level.

2 RESCHEDULING OF MANUFACTURING SYSTEMS

2.1 Uncertainty during schedule execution

Depending on the environment, there may be disruptions during (schedule) execution in the production system, due to unforeseen events, such as machine breakdowns, raw material of insufficient quality or supply, stochasticity of processing times, differences in the operators' efficiency, incorrect or missing information. These are *internal* disruptions and cannot be exactly predicted because of the stochastic behaviour of the parameters, though, reaction from the scheduling system is needed. During execution, incoming urgent orders give the dynamic nature of the scheduling problem and can be concerned as *external* disruptions, which may also require modifications in the schedule.

In order to control production in dynamic scheduling environments having continuous job arrivals or stochastic environments where parameters are uncertain, two common strategies are known. These are dynamic scheduling solutions (on-line scheduling) and predictive-reactive scheduling techniques. The predictive-reactive approach means calculating a predictive (or off-line) schedule concerning a static problem, and continuously updating this existing schedule in order to adapt schedules to changing circumstances (reactive this way).

In the paper, the schedule evaluation techniques related mostly to the predictive-reactive scheduling approach in a dynamically changing environment are discussed, incorporating both deterministic and stochastic system parameters.

Sabuncuoglu and Kizilisik [1] , Vieira, et al. [2], Herroelen and Leus [3] and Gören [4] give a summary in chronologic order of studies that analyze scheduling and rescheduling problems in a dynamic and stochastic environment, while Pinedo [5] categorize the scheduling techniques, based on the stochastic or deterministic characteristics of the problem. Research results on scheduling with uncertainties such as completely reactive, robust scheduling and predictive-reactive approaches are categorized and presented by Aytug et al. [6]. They give a broad overview in their study on production *schedule execution* in the face of *uncertainties*.

2.2 Rescheduling policies

From the practical point of view, it is not possible to create schedules excessively frequently, however, the theoretically best performance of the whole system could be realized if schedules could be adapted to any changes, disruptions occurring in real-time. Most industrial planning and scheduling systems create schedules in idle time of the production, e.g., at nights, since the acquisition of production-related data, definition of constrains and creation of schedules for larger shops, generally requires significant computational time. This way, the basic question "when to reschedule?" needs to be answered.

A notation of existing approaches is provided in [7] and [6]. Let the time when a new schedule is constructed be defined by the rescheduling point and the time between two consecutive rescheduling points by the rescheduling interval (*RI*). Predictive-reactive strategy includes three policy types: periodic, event-driven and hybrid. Schedule modification can be executed in given time periods (*periodic* rescheduling policy) where any events occurring between rescheduling points are ignored up to the following rescheduling point, or related to specified events occurring during schedule execution (*event-driven* rescheduling policy). If this specified event means a disruption or an event that has significant impact on the further schedule execution, then the schedule must be revised or a new schedule must be generated.

Combining the two basic methods, hybrid rescheduling policy can be defined under which rescheduling may occur not only periodically but also whenever a disturbance is realized (either internal or external) in the system (e.g., machine failures, urgent orders). In Figure 1, the concept of hybrid rescheduling policy is presented. Generally, schedules are calculated in every *RI* time interval. Rescheduling is also performed right after Disruption 1 (*RI* is modified to *RI**), while the disruption has significant impact on schedule execution, thus the initial schedule necessitates modification, i.e., rescheduling. Disruption 2 is neglected, because the effect induced by the disruption does not require modification in the schedule, the schedule is still executable without much degradation of performance (e.g. it is not necessary to reschedule, even because Disruption 2 is close to the next rescheduling point).

Continuous rescheduling is the extreme case of event-driven policy where rescheduling action is taken each time an event is recognized by the system.

2.3 Schedule repair methods

Once the system performed the rescheduling action, the way of schedule modification has to be defined. Three common schedule repair methods and robust scheduling are presented below.

Right-shift schedule repair method postpones each operation affected by the disruption by the amount of time needed for making the schedule feasible [8].

Partial rescheduling means that only a selected sub-set of the operations are rescheduled. This method, presented in [1],[8],[9], preserves the initial schedule as much as possible, i.e., only repairs the schedule.

Complete rescheduling in this context means that at each rescheduling point, all jobs from the previous (initial) schedule that remained unprocessed are involved during the schedule formulation. Complete rescheduling is, generally, better than partial rescheduling, regarding efficiency measures.

3 IMPACT OF RESCHEDULING – STABILITY

It might be interesting that while scheduling will optimize the efficiency measure, the applied strategies generate schedules that are often radically different from the previous ones. From the practical point of view, scheduling techniques addressing continuity of schedules during revision seem to be more acceptable or preferable in industrial applications, while constructing completely new schedules during schedule execution process should be avoided.

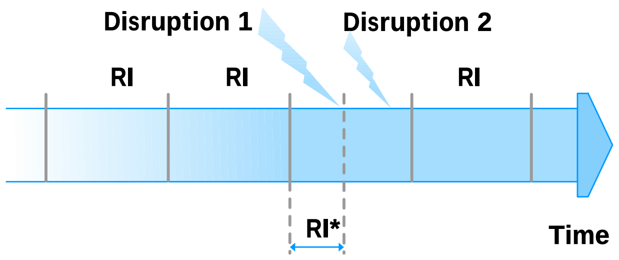

Figure 1: Impact of disruptions on schedule execution by applying hybrid rescheduling policy.

3.1 Schedule stability

It is important to point out that while rescheduling will aim at optimizing efficiency by considering classical performance measures (e.g., makespan or tardiness), the impact of disruptions induced by moving jobs during a rescheduling event is mostly neglected. In case of minimizing this impact the performance measurement applied is frequently called stability [7],[9],[10],. According to Herroelen and Leus [3], if the objective is to generate a new schedule that deviates from the original one as little as possible, a particular rescheduling case is created where ex post stability is induced (contrary to robust scheduling case where, the basic objective is to minimize a function of the deviation between the initial schedule and the final schedule, referred to as ex ante stability).

3.2 Stability measurements and solutions

In previous papers, the number of rescheduling actions was used by Church and Uzsoy [7] as the measure of stability. Alagöz and Azizoglu [11] studied the case in which the stability measure is the number of jobs processed on different machines in the initial and the new schedule. Other approaches [9],[12] defined stability in terms of the deviation of job starting times between the original and revised schedules and the difference of job sequences between the initial and revised schedules.

Starting time deviation can be a very useful measure of the rescheduling algorithm in manufacturing environments where secondary resources (e.g. tooling) are delivered to the machine, based on the initial schedule. Changing job starting times may incur transportation costs if the tools or materials are delivered earlier than required, moreover, rush order costs are also incurred in case these are requested earlier than planned in the schedule. For this reason, in [8] starting time deviation comprises two components: delay and rush.

Measuring sequence deviation is clearly obvious if machine setups are prepared in advance, based on the initial schedule (initial order sequence). For example, if jobs are waiting in sequence queues in order to feed a production line, a sequence change incurs costs in resequencing the queue, reallocate associated resources, etc.

One of the shortcomings of these approaches, i.e., measuring start time, sequence or machine deviations, is that they ignore the fact that the impact of changes increases if those are closer to the current time. In [13] one of the dimensions of stability reflects how close the changes are to the current time (referred to as *actuality* in [14]).

3.3 Multiobjective solutions

Several authors propose a method for dynamic or stochastic scheduling problems, based on a bicriteria objective function that simultaneously considers efficiency and stability, addressing a compromise between them.

A bicriteria objective function is provided in [12],[15] and [9], in order to minimize makespan and deviation from the initial schedule which is measured by the difference between start times and/or the sequence of operations in the initial and revised schedules. During schedule calculation they apply the bicriteria objective function. In some cases the efficiency or stability itself is a combination of the related measures [13]. Results regarding efficiency and stability as well as multiobjective solutions are presented in the following section.

3.4 Impact of stability on schedule efficiency

Vieira, et al. [16] realized the existence of a conflict between avoiding setups (as a measure of stability) and reducing flow time (measure of efficiency). The rescheduling period significantly affects the above objectives, which statement is also concluded in [1],[7],[13],[14] and [15].

In their study, Mehta and Uzsoy [10] and Cowling and Johansson [12] indicate that schedules that are robust to stochastic disturbances can be generated without much degradation of system performance. As it is demonstrated by the evaluation of test problems in [9], introducing the proposed bicriteria objective function, schedule calculation may result significantly more stable schedules, while retaining near-optimal makespans. Bidot et al. [17] conclude that while rescheduling frequency increases with an enhanced sensitivity factor of the rescheduling threshold, the selected performance measure (makespan) improves. Nevertheless, the number of rescheduling is increased, though they do not consider the effect of rescheduling as a matter of stability.

4 STABILITY-ORIENTED EVALUATION OF RESCHEDULING METHODS

4.1 Proposed stability measure

In the previous sections, several possible solutions for measuring schedule stability are discussed (e.g. sequence deviation, number of rescheduling). Here we propose a solution (1), where stability is calculated for each available job in the system during schedule calculation by giving penalty (*PN*) values, using the relation *penalty = starting time deviation + actuality penalty*. Starting time deviation is the difference between the start time of the job at the new and previous rescheduling points. Measuring start time deviation also reflects the changes in the sequences before the machines, i.e. with limitations it is capable to measure the sequence changes.

Actuality penalty is related to a penalty function associated with the deviation of the start time of the job from the current time. For instance, in case two jobs have the same penalty values for start time deviation after rescheduling, the one with the starting time closer to the time of execution will be given a higher penalty value (by applying (1)). Scaling factor (*k*) and the square root operation are responsible for tuning the expression, i.e., for smoothening the *actuality* curve. Penalty values are only calculated in case starting time deviation is greater than 0. A schedule with less penalty value can be considered a more stable schedule. The mean value of stability $\overline{PN}$ is calculated for all schedules as follows:

$$\overline{PN} = \frac{1}{n_{pn}} \sum_{j \in B} \left[\left| t'_j - t_j \right| + \frac{k}{\sqrt{t_j - T}} \right], \qquad (1)$$

where

B is the set of available jobs *j* that remained unprocessed in the initial schedule, and $|t_j' - t_j| > 0$ and $t_j - T > 0$,

n_{pn} is the number of the elements in *B*,

t_j is the predicted start time of job *j* in the current schedule,

t_j' is the predicted start time of job *j* in the successive schedule,

T is the current time, i.e., the point in time at which the rescheduling action is performed,

k is the scaling factor for actuality penalty.

Let us consider a small example here (case1 and case2), demonstrating the calculation of the given penalty values for a single operation after rescheduling. The current time, *T,* when the rescheduling action is taken is 28 and the scaling factor (*k*) is 10. For case1, the start time of the operation is 30 in the current schedule, while in the newly calculated one it equals 32. For case2, these are 45 and 50, respectively. The resulted actuality penalty (highlighted in Figure 2 as *ap1* and *ap2*) is about 7.1 for case1 and only about 2.4 for case2. However the start time deviation for case1 is less as in case2 (2 and 5, respectively), the resulted penalty values (PN in Table 1) show a less critical modification in the schedule of the operation for case2.

	Start time difference	Actuality	Actuality penalty	PN
case1	2	2	7.1	9.1
case2	5	17	2.4	7.4

Table 1: Example - resulted penalty values for one selected operation after rescheduling.

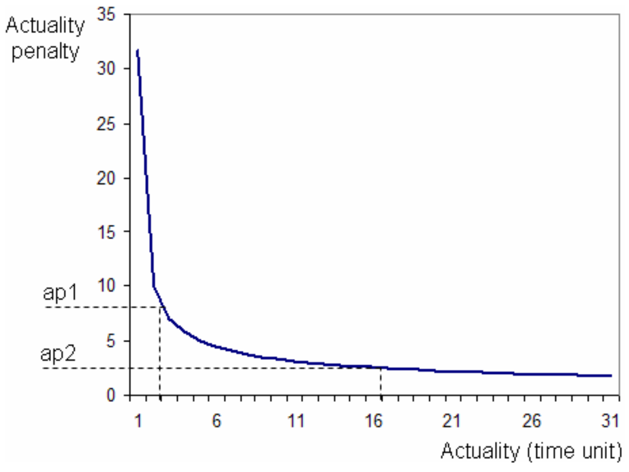

Figure 2: The characteristics of the actuality penalty curve.

4.2 Stability-oriented scheduling by applying GA

The scheduler, connected to the simulation is based on a genetic algorithm (GA).

Since the early ‘80s, several promising GA based scheduling solutions are presented in the scheduling literature, however, the difficulties arising from the chromosome representation for ordering problems (e.g. JSSP) are still inducing potentials in this research field as discussed in [18] and [19]. For instance, the traditional chromosome is usually a binary string. For a scheduling problem, one possible representation is that this string contains the position of each operation in the sequence. The problem with this representation is clear: the applied standard crossover or mutation operators often generate infeasible offspring sequences. One reasonable solution is to use schedule builder or decoder procedures to form a feasible schedule from the chromosome representation.

In our approach we use a problem encoding solution similar to the one presented in [15]. For reducing search space during the generation of active schedules and to be able to use standard GA operators, a modified Giffler-Thompson (G&T) algorithm is presented. The basic concept and proof on the theorem on the active or non-delay schedule creation algorithm can be found in [20], while particular applications in which G&T is hybridized with GA can be found in [15] and [19].

Below a brief outline of the G&T algorithm for obtaining active schedules is presented.

```
Step 1.
    Let $C$ contain the first schedulable op. of each job
    Let $r_{jm} = 0$, for all operation $(j,m)$ in $C$ ($r_{jm}$ is the ready
    time (earliest start time) for the operation $(j,m)$)
Step 2.
    Calculate $t(C) = min_{(j,m) \in C} \{r_{jm} + p_{jm}\}$, ($p_{jm}$ is the processing time of operation $(j,m)$)
    Let $m^*$ denote the machine on which the minimum is achieved
Step 3.
    Let $G$ denote the conflict set of all operations $(j,m^*)$
    on machine m* such that $r_{jm^*} < t(C)$
Step 4.
    Randomly select one op. from $G$ and schedule it
Step 5.
    Delete the operation from $C$, and include its immediate successor in $C$
Step 6.
    Update $r_{jm}$ in $C$ and return to step 2 until all operation
    are scheduled.
```

In our proposed solution, the schedule is formulated by modifying Step 4 in the original G&T algorithm as follows:

Select the operation from *G* with the smallest priority index and schedule it.

Two parallel chromosome sequences are applied to be able to build up schedules also for job-shop problems where one selected operation can be processed on different machines, i.e. machines are capable for different processes.

The first sequence (chrom 1 in Table 2) contains indexes for all the operations to be scheduled with a priority value. This is generated and selected initially from a set defined between 0 and the number of operations in the schedule. Priority indexes for all the schedulable operations are represented by the chromosomes in the GA (Table 2).

The second sequence (chrom 2 in Table 2) also contains indexes for all the operations (in the same order) but this is responsible for selecting the appropriate machine from the set of the available processes required by the operation, e.g., in the example above, it varies btw. 1 and 3.

The parameters for the proposed hybrid GA scheduler are determined after a set of test runs in order to be able to tune the values for the presented scheduling problem sets. Standard setting for the GA are as follows: generation level is 100, population size is 10, crossover probability rate for the applied PMX operator is 0.8, and mutation probability rate is set to 0.1. Selection of the individuals is based on the roulette-wheel selection strategy. The mutation operation is applied to off-spring solutions with a mutation probability: the value of the gene in the selected chromosome for mutation will be altered randomly, by setting the value between 0 and the number of all operations in the schedule for the first sequence and the number of available machines for the second sequence.

As a novel approach, the GA scheduler is capable of creating more stabile (new) schedules by setting the initial population of the GA at the rescheduling point so that the schedule should be built by modifying the final (or best) population of the preceding GA run. Since this procedure reduces the search space [18],[19], and thus the computation effort, as a by-effect, the probability of ending the search in a local minimum or maximum place is higher. As stated in the previous sections, in case of a rescheduling action, the reason of schedule modification is a certain deviation between the initially calculated and the finally executed schedule. Therefore, regarding stability, this kind of shortcomings of GA-based search procedure might be an advantage during the modification of an existing schedule. In our current solution this 'behaviour' of the GA is represented by the rate of the probability, in which measure the new initial population should use the solutions from the preceding ones. In the presented case-study this is set to 0.8.

operation/ job	1,1	2,1	3,1	1,2	2,2	3,2	1,3	2,3
chrom 1.	8	1	6	5	2	3	4	7
chrom 2	1	2	3	2	3	1	3	1

Table 2: Example (3 jobs $\times$ 3 machines) for chromosome representation applied in the GA for schedule formulation.

5 CASE STUDY- HYBRID RESCHEDULING OF A JOB-SHOP

In this section a simulation-based evaluation of the several settings of the proposed rescheduling threshold and the timing of rescheduling are presented. We concentrate on the relationship between the rescheduling threshold and schedule stability as well as efficiency under different scheduling circumstances. The results of the experiments are valuable for future work in this direction (consider, e.g., dynamic job arrivals as additional disturbances in the rescheduling system).

The system to be (re)scheduled is a five-machine test system. Calculated schedules are executed several times, defined by the number of required replications, by using the simulation model of the system. The evaluation of rescheduling capability, i.e., the reactions when realizing disruptions in the proposed scheduler are analyzed. The applied internal disruption-category during schedule execution is machine breakdowns.

By applying discrete event simulation, the solution methods for rescheduling described in the previous sections can be profoundly tested, analyzed and compared in a dynamic, changing test environment. In this section the simulation-based evaluation approaches are presented, while the detailed theoretical background and methodology of simulation itself can be found in [21]. A detailed description of the possible application areas of simulation and new modelling techniques in production planning and scheduling systems are given in [22].

5.1 Design of the rescheduling experiments

Three different scheduling problems are considered, denoted as *alt0*, *alt1* and *alt2*. Case *alt0* is a classical job-shop scheduling problem where each job must be processed on each machine only once and the machines are not interchangeable.

Settings *alt1* and *alt2* cover a job-shop where operations can be processed on different machines, i.e., alternative machines may be selected. The number of alternative machines for each operation (average number of machines per operation) is set to 2 and 3 for problems *alt1* and *alt2*, respectively. The above presented scheduling problems were generated with the purpose to have similar characteristics compared to the benchmark problem sets (s*data, rdata, vdata)* presented by Fisher and Thompson [23] and Hurink et al [24].

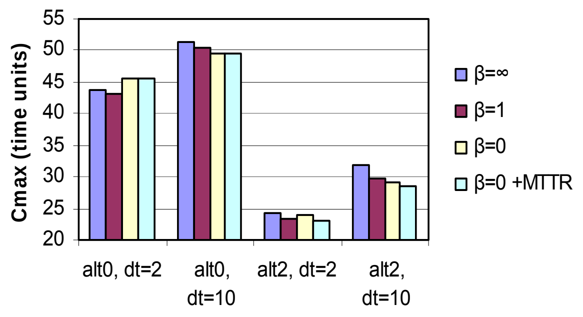

Figure 3: Effect of the duration of the machine breakdown (*dt*) on efficiency.

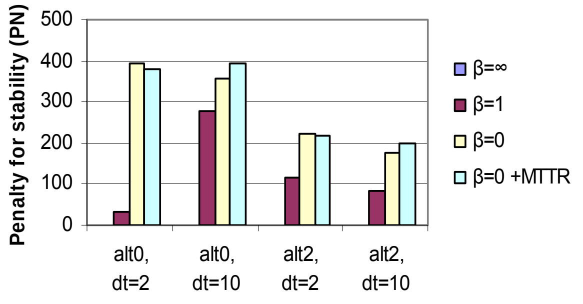

Figure 4: Effect of the duration of the machine breakdown (*dt*) on stability.

The scheduling problem presented can be considered as a *5 X 8* job-shop scheduling problem*,* and this way, the number of machines (*m*) is 5, the number of operations (n_o) per job is $n_o = m$. The efficiency measure for the schedules is makespan calculated $C_{max} = \max\{c_1, c_2, .., c_n\}$, where *n* is the number of jobs. The number of jobs to be scheduled in each initial schedule is 8. Resulted best makespan values, calculated by the GA scheduler concerning deterministic system parameters, static scheduling problem and no machine breakdowns, are 42, 30 and 21 time units, for *alt0*, *alt1* and *alt2,* respectively.

5.2 Rescheduling threshold

In the presented experiments *end time monitoring* is considered as the monitored performance index during the schedule execution. In case the processing of an operation ends, the mean absolute tardiness is calculated for all the operation involved and compared to the *rescheduling threshold* (*β*) by applying (2).

$$\frac{1}{n}\sum_{j=1}^{n}\left|c_{j,sim} - c_{j,pre}\right| > \beta\,, \qquad (2)$$

where

n is the number of the completed operations counted from the last rescheduling point,
$c_{j,sim}$ is the simulated end time of operation *j,*
$c_{j,pre}$ is the predicted end time of operation *j.*

Once the threshold is bypassed, a rescheduling action is initiated and a new schedule is generated. The way of the schedule creation (how to repair the schedule) depends on the settings of the experiment: right-shift or complete rescheduling is applied. In the current static job-shop problem the total penalty values (*PN*) – regarding stability after rescheduling – are calculated by using scaling factor *k* = 10.

Four different settings for the rescheduling strategy are introduced. Setting *β*=∞ can be considered as a right-shift schedule repair after machine breakdown, i.e., no rescheduling action could be performed but each operation affected by the breakdown is delayed by the simulation. By the next setting, *β*= 1 and for the level of *n* 4 is applied, which has a major effect on the reaction time to disruptions (e.g. compared to *n*=0).

β=0 can be considered an event-driven rescheduling method (noted as *β*=0) where each execution-related event, i.e., disruption realized by the system indicates a rescheduling action. A modification of this setting is also introduced. By giving additional information, the mean time to repair (MTTR), about the machine breakdown to the scheduler (*β*=0 + MTTR).

5.3 Evaluation against machine breakdowns

The four different settings for the rescheduling strategy is presented and analyzed on two particular machine breakdown cases as well as on the three scheduling problems (*alt0*, *alt1* and *alt2*). It is assumed that no operation can be processed during the disruption, and job preemptions are not allowed so that disrupted operations must be restarted form the beginning. Each failure generated into the system occurs only once in the scheduling horizon.

By the first set of machine failures, the effect of the breakdown duration is considered. One selected machine is disabled from time unit 5 until time unit 7 and 15, denoted as *dt*=2 and *dt*=10, respectively. Results of the simulation-based evaluation show, in case a particular disturbance occurs in the system, predictive-reactive approach, applying stability-oriented rescheduling methods, a slight improvement on efficiency can be obtained, comparing it with standard right-shifting schedule repair methods (Figure 3 and Figure 4 for the results obtained by the problem set *alt0 and alt2*). The results indicate that if in the system no alternative resources for operations exist (problem set *alt0*), event-driven rescheduling (*β*=0), might be more effective than monitoring rescheduling threshold with a higher value in case *dt* is 10, while keeping stability in an acceptable range. Similarly to the case study presented in [14] (one machine case), increase on the efficiency measure results a degradation of the measured stability can be also observed.

As the second set of breakdowns, the time of the disturbance occurred in the system (relative to the time of execution) is considered. Aggregated results are highlighted

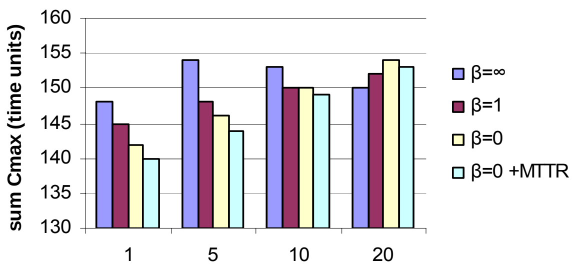

Figure 5: Resulted efficiency measures on the applied rescheduling threshold (β) and the time of breakdowns (T_{br}).

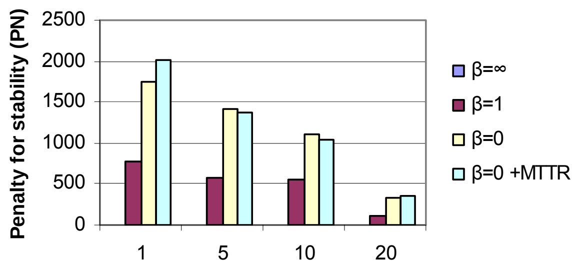

Figure 6: Resulted stability measures on the applied rescheduling threshold (β) and the time of breakdowns (T_{br}).

for problem set *alt0, alt1* and *alt2* in Figure 5 for efficiency and Figure 6 for stability measures. The x-axis represents the time point at which the machine breakdown occurs in the system (1, 5, 10 or 20). It can be stated that right-shift method outperforms the GA-based rescheduling method only in case of the machine breakdown occurs nearly at the end of the considered scheduling horizon (time point 20). However, if the disturbance occurs right after the schedule creation – normal (re)scheduling point – or in the middle of the scheduling horizon, it is obvious to apply the proposed rescheduling method. The selection of the most suitable rescheduling threshold (β) in these cases depends on the required level of stability. For instance, compare the cumulated makespan for the rescheduling threshold settings $\beta=1$ to $\beta=0+MTTR$ at a machine breakdown initiated at time point 5. The values are 148 and 144, respectively; meanwhile the resulted stability (*PN*) values for these settings are 586 and 1362.

6 SUMMARY AND FUTURE WORK

The paper described a concept of schedule stability measurement for rescheduling, as well as a scheduler, based on a GA hybridized with a modified G&T algorithm. By applying the proposed scheduler and executing the resulted schedules by simulation, the solution methods for stability-oriented rescheduling described in the paper were tested and analyzed. It can be stated that the investigated rescheduling parameters (rescheduling threshold) and the disturbance generated into the system had considerable impact on the number of rescheduling actions and the timing of these actions, and by this way, on the stability of the schedule execution. As it is assumed, finding the appropriate rescheduling threshold for each given rescheduling situation may result in a compromise between stabile schedule execution and schedule quality.

The extension of these experiments could be future work by applying adaptive, situation-dependent rescheduling thresholds and stability factors in a dynamic and stochastic scheduling environment. Another direction of the research activity is to improve the applied GA-based scheduler in order to have better performance both on efficiency and stability, e.g., by applying multi-objective objective function.

7 ACKNOWLEDGMENT

The research was supported, partially, by the project ‘Real-time, cooperative enterprises’ in the frame of the National Research and Development Programme by the Ministry of Education, Hungary (Grant No. 2/010/2004). A part of the work was covered by the Hungarian Scientific Research Fund, Hungary, Grant Nos. T043547 and T049481. Botond Kádár greatly acknowledges the support of Bolyai János Scholarship of the Hungarian Academy of Sciences, significantly easing the contribution.

8 REFERENCES

[1] Sabuncuoglu, I., Kizilisik, O.M., 2003, Reactive Scheduling in a Dynamic and Stochastic FMS Environment, Int. J. of Prod. Res., 41/17: 4211-4231.

[2] Vieira, G.E., Herrmann, J.W., Lin, E., 2003, Rescheduling Manufacturing Systems: A Framework of Strategies, Policies and Methods, Journal of Scheduling, 6/1: 39-62.

[3] Herroelen, W., Leus, R., 2005, Project Scheduling Under Uncertainty: Survey and Research Potential, European J. of Operational Res., 165/2: 289-306.

[4] Gören, S., 2002, Robustness and Stability Measures for Scheduling Policies in a Single Machine Environment, MSc Thesis, Department of Industrial Engineering, Institute of Engineering and Sciences, Bilkent University, Turkey.

[5] Pinedo, M., 2002, Scheduling: Theory, Algorithms and Systems, Prentice Hall, NJ, USA.

[6] Aytug, H., Lawley, M.A., McKay, K., Mohan, S., Uzsoy, R., 2005, Executing Production Schedules in the Face of Uncertainties: A Review and Some Future Directions, Europ. J. of Op. Res., 161: 86-117.

[7] Church, L.K., Uzsoy, R., 1992, Analysis of Periodic and Event-Driven Rescheduling Policies in Dynamic Shops, Int. J. of Computer Integrated Manufacturing, 5: 153-163.

[8] Abumaizar, R.J., Svestka, J.A., 1997, Rescheduling Job Shops Under Random Disruptions, International Journal of Production Research, 35: 2065-2082.

[9] Wu, S.D., Storer, R.H., Chang, P.C., 1993, One-Machine Rescheduling Heuristics with Efficiency and Stability as Criteria, Comp. and Op. Res., 20: 1-14.

[10] Mehta, S.V., Uzsoy, R.M., 1998, Predictable Scheduling of a Job Shop Subject to Breakdowns, IEEE Trans. on Robotics and Automation, 14: 365-378.

[11] Alagöz, O., Azizoglu, M., 2003, Rescheduling of Identical Parallel Machines Under Machine Eligibility Constraints, European J. of Op. Res., 149: 523–532.

[12] Cowling, P., Johansson, M., 2002, Using Real Time Information for Effective Dynamic Scheduling, European J. of Operational Research, 139: 230-244.

[13] Rangsaritratsamee, R., Ferrell, Jr., W.G., Kurz, M.B., 2004, Dynamic Rescheduling that Simultaneously Considers Efficiency and Stability, Computers & Industrial Engineering, 46: 1-15.

[14] Pfeiffer, A., Kádár, B., Csáji, B.Cs., Monostori, L., 2005, Simulation Supported Analysis of a Dynamic Rescheduling System, in: Manufacturing, Modelling, Management and Control 2005, edited by Chryssolouris, G. and Mourtzis, D., Elsevier: 25-30.

[15] Leon, V.J., Wu, S.D., Storer, R.H., 1994, Robustness Measures and Robust Scheduling for Job Shops, IIE Transactions, 26/5: 32–43.

[16] Vieira, G.E., Herrmann, J.W., Lin, E., 2000, Predicting the Performance of Rescheduling Strategies for Parallel Machines Systems, Journal of Manufacturing Systems, 19: 256-266.

[17] Bidot, J., Laborie, P., Beck, J.C., Vidal, T., 2003, Using Simulation for Execution Monitoring and On-Line Rescheduling with Uncertain Durations, In: Proc. of the ICAPS'03 Workshop on Plan Execution, Trento, Italy.

[18] Bierwirth, C., Mattfeld, D.C., 1999, Production Scheduling and Rescheduling with Genetic Algorithms, Evolutionary Computation, 7/1: 1-17.

[19] Park, B.J., Choi, H.R., Kim, H.S., 2003, A Hybrid Genetic Algorithm for the Job Shop Scheduling Problems, Comp. and Industrial Engineering, 45: 597-613.

[20] Giffler, B., Thompson, G.L., 1960, Algorithms for Solving Production-Scheduling Problems, Operations Research, 8: 487–503.

[21] Law, A., Kelton, D., 2000, Simulation Modelling and Analysis, McGraw-Hill.

[22] Kádár, B., Pfeiffer, A., Monostori, L., 2004, Discrete Event Simulation for Supporting Production Planning and Scheduling Decisions in Digital Factories, In: Proc. of the 37th CIRP Int. Seminar on Manufacturing Systems; Digital Enterprises, Production Networks, Budapest, Hungary: 444-448.

[23] Fisher, H., Thompson, G.L., 1963, Probabilistic Learning Combinations of Local Job Shop Scheduling Rules, In: Industrial Scheduling, Muth, J.F., Thompson, G.L., eds., Prentice Hall, Englewood Cliffs.

[24] Hurink, E., Jurisch, B., Thole, M., 1994, Tabu Search for the Job Shop Scheduling Problem with Multi-Purpose Machine, Op. Res. Spectrum, 15: 205-215.