International Journal of Production Research,
Vol. 43, No. 15, 1 August 2005, 3103–3129

Taylor & Francis
Taylor & Francis Group

Anticipation and flexibility in dynamic scheduling

JÜRGEN BRANKE*† and DIRK CHRISTIAN MATTFELD‡

†Institute AIFB, University of Karlsruhe, 76131 Karlsruhe, Germany
‡Department for Business Computing, University of Braunschweig,
38106 Braunschweig, Germany

(*Revision received January 2005*)

Many real-world optimization problems change over time and require frequent re-optimization. We suggest that in such environments, an optimization algorithm should reflect the problem's dynamics and explicitly take into account that changes to the current solution are to be expected. We claim that this can be achieved by having the optimization algorithm search for solutions that are not only good, but also flexible, i.e. easily adjustable if necessary in the case of problem changes. For the example of a job-shop with jobs arriving non-deterministically over time, we demonstrate that avoiding early idle times increases flexibility, and thus that the incorporation of an early idle time penalty as secondary objective into the scheduling algorithm can greatly enhance the overall system performance.

Keywords: Job-shop scheduling; Rescheduling; Manufacturing flexibility; Evolutionary algorithm; Dynamic environment; Priority rules

1. Introduction

Although most literature on optimization assumes static, fully defined problem instances, many real-world optimization problems are dynamic and stochastically change over time. For example, in manufacturing, new orders arrive frequently and have to be integrated into the schedule. Owing to the current trend towards just-in-time manufacturing, and technological progress increasingly facilitating the processing of real-time data, dynamically changing optimization problems can be encountered more and more frequently, and seem to become the norm rather than the exception. Therefore, there is a need for optimization heuristics that can handle such quickly changing environments.

One standard approach to deal with a non-deterministic problem is to constitute and solve a new deterministic sub-problem every time the problem data change. However, even if each sub-problem is solved to optimality, the overall solution is unlikely to be optimal. Considering the sub-problems as being completely independent disregards the impact a solution may have on the system's state, and thus on the problems encountered in the future. In this paper, we make the following conjecture: if a problem requires sequential decision making under an uncertain future, and if

*Corresponding author. Email: branke@aifb.uni-karlsruhe.de

International Journal of Production Research
ISSN 0020–7543 print/ISSN 1366–588X online
http://www.tandf.co.uk/journals
DOI: 10.1080/00207540500077140

the decisions impact the future state of the system, decision making should anticipate future needs. This means that an optimization algorithm should not just focus on the primary objective function, but should additionally try to move the system into a flexible state, i.e. a state that facilitates adaptation if necessary.

We are going to examine this conjecture, considering a dynamic total tardiness job-shop scheduling problem with new jobs arriving non-deterministically over time. Clearly, the scheduling decisions influence the shop floor's future state, e.g. its work in process, the interdependencies between jobs, and the distribution of machine idle times. As we will demonstrate, the flexibility in a job-shop is largely determined by its machine capacity and can be effectively preserved by avoiding early machine idle times in the forthcoming schedule. As a consequence, we propose to introduce a second objective into the scheduling algorithm: a penalty for early idle times, henceforth also called 'flexibility term'. The resulting algorithm is denoted as 'anticipatory scheduling', because the integrated flexibility term implicitly anticipates the need to incorporate new jobs in the future. We show empirically that anticipatory scheduling can greatly improve the system's performance with respect to the original tardiness objective when applied in a dynamic non-deterministic[1] shop-floor environment.

Note that our approach is largely independent of the optimization method applied, as the flexibility term only modifies the objective function, away from an exclusive focus on tardiness of a single deterministic sub-problem towards a compromise between tardiness and flexibility. This is done in order to facilitate the transition between the current and the next deterministic subproblem. In our paper, we integrate the flexibility term into biased random sampling (BRS), a randomized priority rule-based approach which is shown to outperform traditional rule-based approaches like SPT or RR. Additionally, we engage an evolutionary algorithm (EA), which allows us to replace the non-delay scheduling mechanism of BRS by a more elaborate mechanism which creates active schedules. This latter approach is particularly interesting in so far, as active scheduling introduces additional idle times—which is at the same time penalized by the flexibility term in the modified objective function. We are going to show that the overall tardiness can benefit despite this apparent contradiction.

The paper is organized as follows. We start in section 2 with a discussion of some related work. In section 3, we define the dynamic non-deterministic job-shop problem that serves as a test-bed for our approach. The main ideas and basic results are described in sections 4–6. First, in section 4, we reason about scheduling flexibility and present our approach of penalizing early idle times. This approach is then incorporated into BRS (section 5) and tested in section 6. Sections 7–12 examine the robustness and general applicability of anticipatory scheduling. Section 7 looks at some alternative weighting schemes, section 8 uses an evolutionary algorithm instead of BRS, and section 9 compares non-delay with active scheduling. A comparison of the presented anticipatory scheduling approaches with the best-known

[1]Unfortunately, there exist different terminologies to describe changing problems. While many would consider a problem changing over time as 'dynamic', in the scheduling community, such a problem is generally named 'stochastic' or 'non-deterministic'. We use both terms in this paper, depending on the context.

priority rule can be found in section 10, the issue of schedule nervousness is examined in section 11, and section 12 considers some additional problem instances. The paper concludes with a summary and some ideas for future work.

2. Related work

The flexibility of a firm's manufacturing system has major implications on the firm's competitive strength (e.g. Gupta and Somers 1992). Intuitively, the term flexibility describes a system's ability to adapt to a dynamic environment (Pereira and Paulré 2001). A more formal definition, however, seems difficult. Several authors have suggested a wide range of different definitions; a survey can be found, for example, in Gupta and Somers (1992). Recently, Chang *et al*. (2001) pointed out that all these definitions can be subsumed by two underlying aspects: versatility and efficiency.

Versatility expresses the system's capability of performing a wide range of tasks, whereas efficiency refers to how fast the manufacturing system operates in performing these tasks. While versatility and flexibility may be used almost synonymously, the relation between efficiency and flexibility is not equally obvious. But efficiency increases throughput and reduces work in process inventory, which in turn provides the leeway often necessary to adapt quickly to changing conditions.

The impact of scheduling on manufacturing flexibility has been investigated predominantly by case studies. Persentili and Alptekin (2000) compare two manufacturing control strategies, namely MRP-push and JIT-pull, with respect to efficiency measures as well as flexibility concerning product type variations. Flexibility in the sense of being able to quickly satisfy additional orders of the same type can also be obtained by large inventories. Pagell *et al*. (2000) look at the balance between system flexibility and inventory buffers while satisfying a global performance measure.

For scheduling in the case of dynamically arriving jobs, taking into account more information usually improves performance. According to Stadler (2005), the planning horizon should at least cover an interval of time which corresponds to the flow time of the largest job considered. Information about the future is incomplete, and as new information becomes available, the current schedule and its associated quality will be challenged. There are two remedies for this dilemma: one is to accept frequent rescheduling, while the other is to renounce planning completely and delaying decisions to the very last moment.

Scheduling on the basis of a rolling-time horizon falls into the first category: at certain points in time, deterministic sub-problems are generated with respect to the current state of the system. For each sub-problem, only the front part of the generated schedule is actually implemented; the latter and typically larger part of the schedule is subject to rescheduling in subsequent problems. In order to solve each deterministic problem instance, typically scheduling algorithms originally developed for static problems are employed. However, solving each sub-problem to myopic optimality will usually not lead to the globally optimal solution (Baker 1997, Powell *et al*. 1998).

For an early investigation of scheduling with a rolling time horizon on a total tardiness job-shop problem, see Raman and Talbot (1993). The shorter the rescheduling interval is chosen, the more responsive the approach becomes as information

is taken into account earlier. Previous work (Farn and Muhlemann 1979, Adam and Surkis 1980, Muhlemann *et al*. 1982) suggests that an increasing rescheduling frequency can improve the scheduling performance. Yamamoto and Nof (1985) propose event-triggered rescheduling, i.e. deterministic sub-problems are created as soon as new jobs arrive in the manufacturing system. Church and Uzsoy (1992) demonstrate the advantages of event triggered rescheduling over interval-based rescheduling. Fang and Xi (1997) propose a combined technique which performs event-based rescheduling in case of machine breakdowns and interval-based rescheduling to cope with the arrival of new jobs.

If either the problem changes so quickly that the required frequency for generating global schedules becomes prohibitive, or the problem changes so drastically that no meaningful information can be obtained from pre-schedules, priority-based dispatching offers a serious alternative. Because only up-to-date local information is taken into account, priority-based dispatching is an obvious candidate for dynamic scheduling problems (Morton and Pentico 1993). However, the success of rule-based dispatching is limited by the lack of a global objective function. As a remedy, Kutanoglu and Wu (1998) suggest combining global optimization with local dispatching by determining job priorities on a global level beforehand, while deferring the priority driven dispatching as long as possible. Rajendran and Holthaus (1999) provide an extensive comparison of several dispatching rules for dynamic stochastic job-shops. Pierreval and Mebarki (1997) suggest selecting appropriate dispatching rules depending on the current system state.

A side-effect of re-scheduling is the disruption inflicted upon the current schedule, often called 'nervousness'. Such disruptions may be costly, e.g. because of internal change costs, or because of now violated agreements with customers and suppliers based on the original schedule. Although nervousness is not our primary concern, we will report here also on how our algorithm performs in comparison which others with respect to nervousness. For papers explicitly aimed at rescheduling cost due to new orders, see Hall and Potts (2004), who provide algorithms to minimize rescheduling cost on different single machine problems, or Li *et al*. (1993), and Jain and Elmaraghy (1997), who restrict re-scheduling to a subset of operations affected.

None of the approaches described so far anticipates future changes, although the importance of anticipation has already been realized in Kimenmia and Gershwin (1983). Anticipating future events is even a necessity when searching for robust schedules, i.e. when a schedule's *expected* quality in an uncertain environment is optimized. Examples include the work by Leon *et al*. (1994), who derive an explicit measure for robustness, or Jensen (2001), who uses a stochastic evaluation as part of an evolutionary algorithm. Mehta and Uzsoy (1998) produce robust schedules by inserting idle times in anticipation of machine breakdowns. Following a similar idea, Yellig and Mackulak (1997) propose a method called 'capacity hedging', reserving a certain proportion of the capacity such that in the case of machine breakdowns, production can be accelerated to catch up with the interrupted schedule. But while robustness is required for solutions that are assumed to persist unchanged, in this paper we seek flexibility, i.e. we assume the necessity of repeated adaptations of a solution (Branke and Mattfeld 2000, Branke 2001). These two aspects are not necessarily unrelated, as Jensen (2001) has observed that robust solutions are often flexible as well.

In an earlier paper (Branke and Mattfeld 2000), we have proposed penalizing early machine idle-times in a rescheduling algorithm's objective function in order to maintain flexibility with respect to the arrival of future jobs. This approach tends to exhaust machine capacity and withholds machine idle time in anticipation of future demand. In that sense, it could also be seen as a form of capacity hedging, but by using *more* capacity earlier in the schedule than the original goal criterion would suggest. In this paper, we discuss our approach much more thoroughly, with more empirical evidence and from a more general perspective. We show that anticipatory scheduling provides excellent scheduling performance in an inexpensive and immediate way.

3. Dynamic job-shop scheduling

We consider a dynamic job-shop problem where n jobs are to be processed on m machines. The processing of a job on a certain machine is referred to as operation. Processing times are deterministic, and the processing times of operations belonging to job i add up to the job's total processing time p_i. Jobs become known non-deterministically over time, which implies that job i cannot start before its arrival at release time r_i. Every job is processed in a prescribed technological order, which does not necessarily cover all machines, but no machine more than once. A job can be processed by one machine at a time only, one machine can process just one job at a time, and preemption is not allowed. Due dates d_i indicate the point in time at which a job should be completed; the actual time c_i of completing job i, however, is subject to search. The spread between the actual and desired completion time is taken into account by minimizing the mean tardiness $\bar{T} = (1/n) \sum_{i=1}^{n} \max\{c_i - d_i, 0\}$.

For generating the dynamic problem, we refer to an experimental setting often used for simulating manufacturing systems; cf. Holthaus and Rajendran (1997). The interarrival times of jobs in the manufacturing system affect its workload, i.e. the number of operations in the system which await processing. The mean inter-arrival time λ can be determined by dividing the mean processing time of jobs $\bar{p}$ by the number of machines m and a desired utilization rate U, i.e. $\lambda = \bar{p}/(mU)$. A utilization rate of $U = 0.7$ represents a relaxed situation of the manufacturing system. A moderate load is produced by $U = 0.8$, whereas a utilization rate of $U = 0.9$ corresponds to a heavy workload.

We simulate a simplified manufacturing system by using the following attributes:

- the manufacturing system consists of six machines;
- each job passes four to six different machines, resulting in five operations on average;
- technological orders are generated from a uniform probability distribution;
- processing times of operations are integers and uniformly distributed in the range of [1,19] resulting in a mean processing time of $\bar{p} = 5 \cdot 10$;
- relatively tight due dates are generated by $d_i = r_i + 2 \cdot p_i$; and
- inter-arrival times are exponentially distributed with mean λ.

Besides the above standard setting, we also run some experiments with more flexible deadlines, generated randomly from $[r_i + p_i, r_i + 3p_i]$, and 12 instead of six machines. Details can be found in section 12.

In order to obtain statistical significance, all results are averaged over a number of simulation runs. Note that in the case of dynamic problems, accuracy can be gained either by increasing the running time of a simulation, or by increasing the number of simulations used for averaging. According to Rajendran and Holthaus (1999), given a fixed total simulation time, it seems best to perform few replications of longer runs. Following this suggestion, we generated 10 problem instances with 2200 jobs each. Initially, each schedule is empty. Since we are particularly interested in the steady-state performance, for final evaluation we discard the tardiness associated with the first 100 and last 100 jobs arriving, i.e. our objective function is calculated as $\bar{T} = (1/2000) \sum_{i=101}^{2100} \max\{c_i - d_i, 0\}$. This has been shown to suffice to avoid irregularities during the startup and termination phases of a simulation run (Bierwirth and Mattfeld 1999).

4. A flexibility measure for job-shop scheduling problems

In this section, we motivate and present our approach to implement anticipation in the case of dynamic scheduling. We assume that the dynamic problem is solved by means of a rolling time horizon, i.e. the problem is decomposed into a series of deterministic sub-problems. Whenever a new job arrives at the stochastically determined time t, operations that started before t are considered as implemented and are consequently discarded from further consideration (cf. figure 1). Since the processing of operations is non-preemptive, the set of already implemented operations includes those that are currently being processed. Operations with starting times greater than t

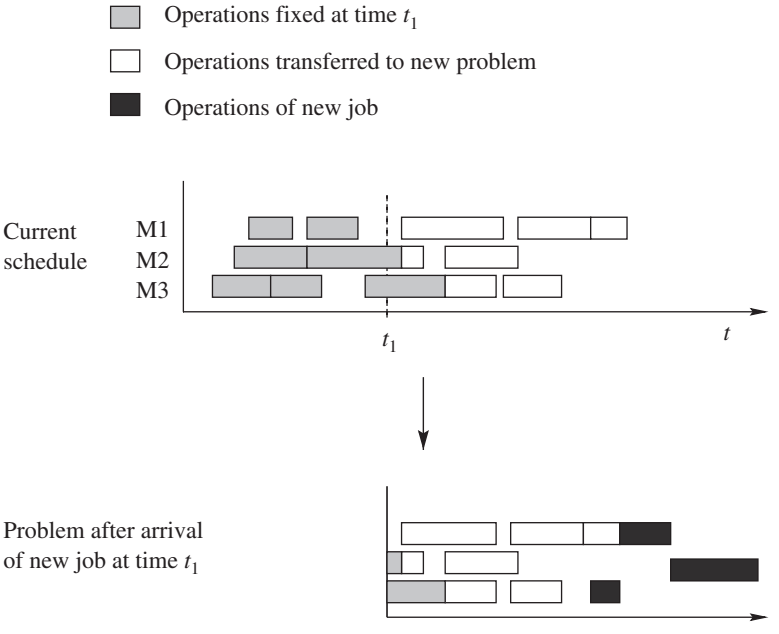

Figure 1. When a new job arrives, a new deterministic scheduling problem is generated, consisting of all operations not yet started plus the operations of the new job.

are subject to rescheduling in the next deterministic sub-problem. For this new problem to be constructed, release times of jobs r_i are re-set to t or, in case of a currently processed operation, to its prospective completion time. Additionally, the time of the earliest availability of machines a_j ($1 \leq j \leq m$) is re-set to t or, if it is currently busy, to the completion time of the associated operation. Finally, the operations of the newly arrived jobs are added to the new sub-problem.

According to the conjecture made in the introduction, schedules derived in a rolling time fashion should not only pursue the primary objective of minimizing tardiness, but also take into account a flexibility measure. The flexible schedules thus generated should be easier to adapt after the arrival of additional jobs, resulting in an improved performance over time.

Consider the example depicted in figure 2: let the due date for both jobs be at $t = 9$, i.e. the two alternative schedules (a) and (b) yield the same tardiness of 1 time unit. Nevertheless, the schedules differ in their distribution of machine utilization. When a new job arrives, it is likely that schedule (b) will be able to integrate it more easily, as machine M2 becomes available earlier than in schedule (a). Thus, by taking into account the times when the operations are processed, the system's flexibility may be increased.

With this introductory example in mind, we are going to reason what measurable aspects of a manufacturing system's state impact the system's flexibility, before we are going to integrate these aspects by modifying the objective function of the scheduling algorithm.

There are some important differences between static and dynamic scheduling: in the static case, the problem is fully defined, and the solution quality is fully described by the objective function criterion under consideration. In other words, machine idle times are inserted regardless of their position early or late in the schedule as long as that helps to achieve a better solution with respect to the given primary objective function.

On the other hand, in the dynamic case, the overall solution quality depends not only on the schedule of the current deterministic sub-problem but also on the

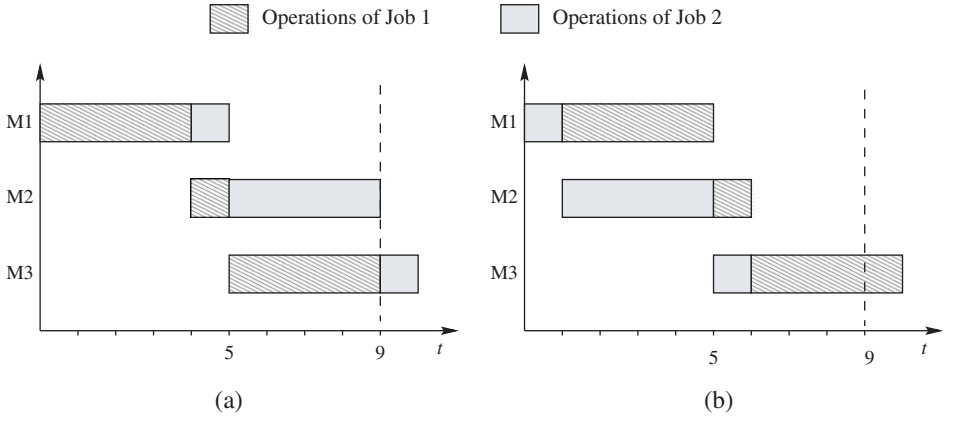

Figure 2. Assuming the performance of both schedules being equal, the schedule depicted in Gantt chart (b) may be favoured because it preserves the idle time of machine M2 longer.

solution quality obtained (or obtainable) in all subsequent sub-problems, with each sub-problem being linked to the next by the transferred backlog. The operation of a certain job on a machine may be re-scheduled many times, but operations scheduled early have a higher probability of being actually implemented before the next change is encountered than operations scheduled later. The probability that a new job will have arrived before time t in the future, causing a re-scheduling, is increasing with t. Hence, near time events are more important: all operations completed before the next arrival of a new job do not have to be re-considered as backlog, thus reducing the sub-problem to be solved in the next stage. A similar argument holds for idle times: while the overall workload for a sub-problem is fixed, any idle times in the early part of the schedule are likely to be implemented, and cannot be used for re-scheduling. On the other hand, any idle time in the latter part of the schedule is likely to facilitate the integration of forthcoming jobs. In that sense, late idle times (or, equivalently, the avoidance of early idle times) correspond to the flexibility desired and mentioned in our conjecture.

We take a rather pragmatic approach to implement the above ideas: in addition to the original objective function (the mean tardiness of jobs $\bar{T}$), we introduce a flexibility term P, penalizing idle times. Since we expect early idle times to be of particular importance for dynamic scheduling, the penalty for a particular idle time is weighted linearly decreasing with the time t of its occurrence in the schedule, i.e. the weight can be calculated as

$$w(t) = \max\left\{0, 1 - \frac{t}{\beta}\right\}.$$

This evaluates idle-time at time $t=0$ with weight 1.0 and idle time at time $t \geq \beta$ with weight 0.0, where β is a user-defined parameter denoting the length of the interval considered. We calculate P by summing up weighted idle times over the machines $1 \leq j \leq m$.

Also, we need to decide to what extent the avoidance of early idle time should replace the original objective function. An appropriate scaling of $\bar{T}$ against P can hardly be derived analytically. We therefore assume to be given a set S of $N = |S|$ schedules which can be used to normalize T and P independently to the interval [0,1] on the basis of minima and maxima observed in S. The objective value f_k of schedule k ($1 \leq k \leq N$) is constructed from a convex combination of both normalized terms, with the parameter α being the weighting factor:

$$f_k = (1-\alpha)\hat{T}_k + \alpha\hat{P}_k \tag{1}$$

with

$$\hat{T}_k = \begin{cases} \dfrac{\bar{T}_k - \min_l\{\bar{T}_l\}}{\max_l\{\bar{T}_l\} - \min_l\{\bar{T}_l\}} & : \quad \max_l\{\bar{T}_l\} > \min_l\{\bar{T}_l\} \\ 0 & : \quad \text{otherwise} \end{cases}$$

$$\hat{P}_k = \begin{cases} \dfrac{P_k - \min_l\{P_l\}}{\max_l\{P_l\} - \min_l\{P_l\}} & : \quad \max_l\{P_l\} > \min_l\{P_l\} \\ 0 & : \quad \text{otherwise.} \end{cases}$$

With $\alpha = 0.0$, the schedule's idle time is not considered at all, whereas with $\alpha = 1.0$ the tardiness does not contribute to f_k. Note that for $\alpha > 0$, this approach

suggests implementing not the best schedule according to tardiness, but one with a possibly higher tardiness and a greater flexibility. The conjecture is that although we select inferior schedules with respect to the original objective function for each of the deterministic sub-problems, this will reduce the tardiness of the complete solution to the entire dynamic problem.

The remainder of this paper is devoted to a computational investigation on the role of machine idle times in dynamic scheduling. In particular, we verify empirically that penalizing early idle times can improve scheduling performance significantly. In principle, our approach should be independent of the actual heuristic search method used, as long as the method generates several alternatives necessary for the normalization of the objectives. In the following section, we will use a randomized rule-based method for heuristic search. A more elaborate evolutionary algorithm will also be investigated in section 8.

5. Dispatching and biased random sampling

Dispatching rules assign priorities to operations that can be scheduled, and these, can then be used by a non-delay schedule builder to decide upon the operations to be dispatched next. Among the large number of priority rules proposed during the last decades, the following three rules are known to reduce the mean job tardiness effectively (Haupt 1989, Rajendran and Holthaus 1999).

- The 'shortest processing time' rule (SPT) gives priority to the operation with the shortest imminent processing time. Jobs waiting in a queue may cause their dedicated successor machine to run idle. SPT alleviates this risk by reducing the length of the queue in the fastest possible way. Therefore, SPT is generally suited for flow time and tardiness-related 'mean objectives', and it is particularly suited under highly loaded shop-floor conditions.
- The slack of job i is defined as the time span left within its allowance $d_i - r_i$, assuming that the remaining operations p_i are performed without any delay. Since jobs may wait in front of each machine, the rule 'slack per number of operations remaining' (S/OPN) favours the job with the minimum ratio of slack and the number of remaining operations. S/OPN is known to reduce the mean tardiness effectively under relaxed shop-floor conditions.
- The 'cost-over-time' rule COVERT combines ideas of SPT and S/OPN. It prioritizes jobs according to the largest ratio of the expected job tardiness (forecast from the current system state) and the operation's processing time. In this way, COVERT retains SPT performance but respects due dates if jobs are late (Russel *et al.* 1987). COVERT has been shown to perform well for varying load conditions and in the event that flow time and tardiness-related objectives are pursued simultaneously.

A schedule builder produces a single schedule on the basis of a certain priority rule. Since we need several alternative schedules, we use each of the above three priority rules in a randomized fashion, a method called biased random sampling (BRS) (cf. Baker 1974, Drexl 1991). At each step of the schedule builder, priorities are calculated for all schedulable operations with respect to the priority rule. Different to the deterministic priority rule-based scheduling, where the operation

with the maximum priority is chosen for dispatching, in BRS the operation to be dispatched is drawn probabilistically in proportion to its priority. In our implementation, a 'roulette wheel selection' as described by Goldberg (1989), is used. In this way, many different schedules can be produced from a single priority rule.

BRS fits our requirements for a schedule of reasonable quality for a deterministic sub-problem, and the set of schedules generated provides the basis for a normalization of T and P according to equation (1). Furthermore, BRS performs well independent of the problem size, certainly better than SPT (as SPT is an integral part of BRS). As we will show in section 10, it also performs significantly better than the RR rule, a dispatching rule which has been shown to be particularly suited for dynamic total tardiness scheduling problems. For each deterministic sub-problem, we generated 49 998 schedules, 16 666 based on each one of the three priority rules described above. Finally, these almost 50 000 schedules are evaluated according to equation (1), and the best schedule is actually implemented (up to the arrival time of the next job). The running time of the heuristic for a typical sub-problem with 40 operations is 2.5 s on a 1.2 GHz PC.

6. Balance between tardiness and flexibility

In this section, we evaluate the tardiness improvements obtainable by the suggested integration of an additional flexibility term into the objective function. We examine the robustness of the approach with respect to several parameters and also show how these affect the work in process (WIP), i.e. the number of operations in the production system. Intuitively, a job-shop with little WIP is more flexible in completing a new job on time. We see that WIP is indeed closely related to the flexibility of the job-shop.

Our approach is parameterized by the length of the time interval considered (β) and the emphasis on idle time (α). In order to examine the interaction between α and β, as well as their impact on the system performance under different load conditions, we run experiments for three different load conditions $U \in \{0.7, 0.8, 0.9\}$, in each case varying $\alpha \in \{0.000, 0.125, \ldots, 1.000\}$ and $\beta \in \{10, 30, \ldots, 150\}$. The largest β value of 150 corresponds approximately to the mean time of at least 10 job arrivals for $U \geq 0.7$, which ensures that the considered time spans several re-scheduling events. For each combination of U, α, and β, the average over the 10 different problem instances described in section 3 is reported. Altogether, a total of $3 \cdot 7 \cdot 8 \cdot 10 = 1680$ runs are performed.

The observed results are shown in figure 3; for a load of $U = 0.8$, details are additionally depicted in table 1. In order to focus on the mean tardiness while varying α and β, the percentage of improvement (reduction) over $\bar{T}$ observed for $\alpha = 0.000$ is reported, i.e. for the reference parameterization, idle times are not considered at all, and consequently the setting of β is of no matter. The plots on the left side of figure 3 show iso-lines in steps of 5% for $U = 0.7$, $U = 0.8$, and $U = 0.9$. Since β has no influence for $\alpha = 0.000$, all solutions along the y-axis represent the reference setting with zero improvement.

The shapes of the iso-line plots resemble each other for the different U considered. The relative improvements that can be obtained by penalizing early idle times are impressive and lie around 15%. The maximal *absolute* improvements

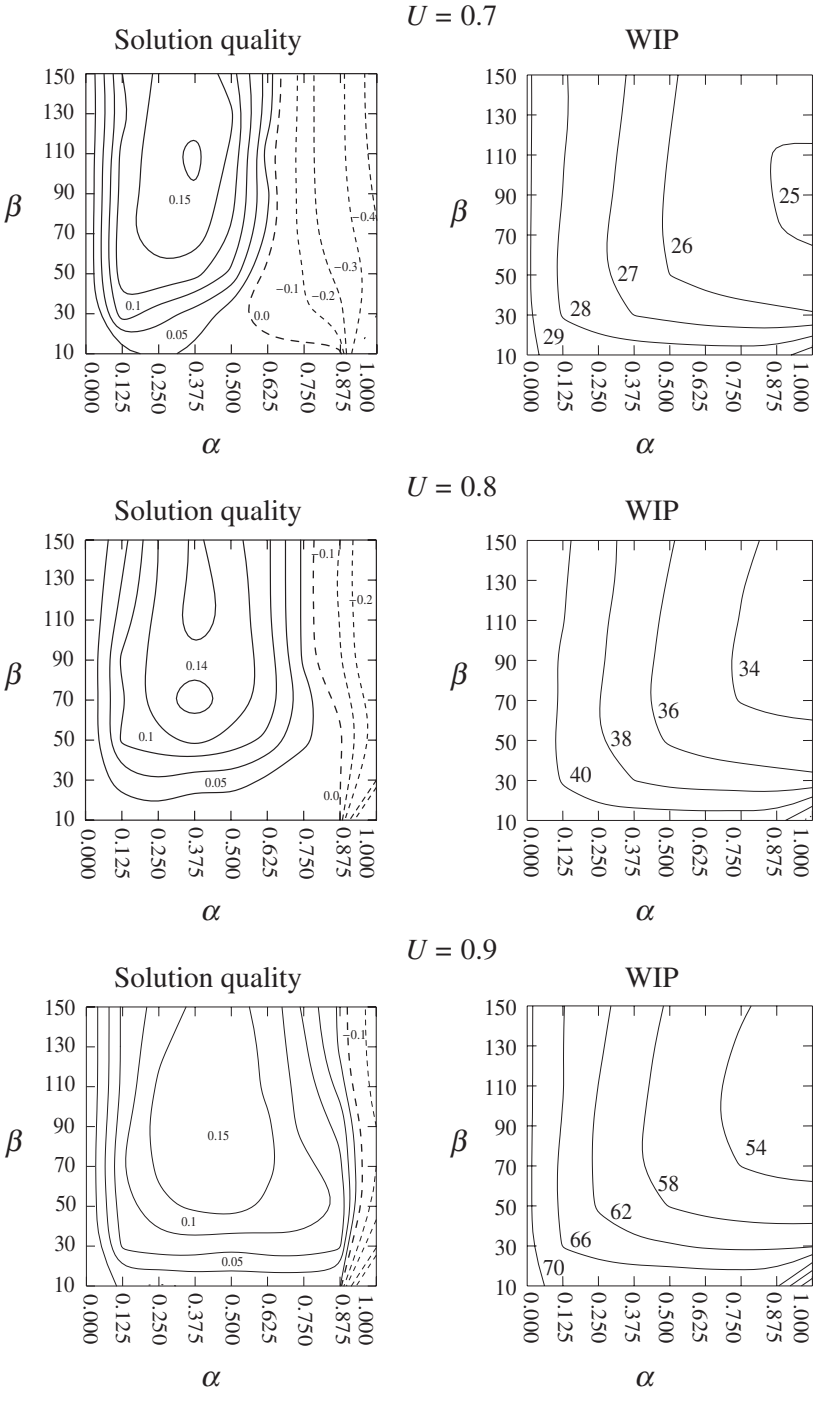

Figure 3. Effect of α and β on the improvement of the solution quality $\bar{T}$ in percent over the result obtained for $\alpha = 0.0$ (left), and the work in process (WIP) depicted as the average number of operations involved in deterministic sub-problems (right). Idle times occurring in $[0, \beta]$ are penalized with linearly decreasing weight.

Table 1. Percentage of $\bar{T}$ improvement achieved over standard BRS ($\alpha = 0$), depending on α and β for $U = 0.8$.

	α							
β	0.125	0.250	0.375	0.500	0.625	0.750	0.875	1.000
150	0.04	0.10	0.13	0.11	0.08	0.02	−0.05	−0.19
130	0.05	0.10	0.13	0.12	0.08	0.02	−0.06	−0.19
110	0.06	0.11	0.14	0.12	0.08	0.02	−0.05	−0.18
90	0.08	0.11	0.12	0.12	0.09	0.02	−0.05	−0.18
70	0.07	0.12	0.14	0.12	0.09	0.04	−0.01	−0.14
50	0.08	0.09	0.11	0.09	0.07	0.03	0.01	−0.12
30	0.04	0.05	0.03	0.04	0.02	0.01	0.00	−0.20
10	0.00	0.01	0.01	0.00	0.01	−0.00	−0.00	−0.76

in tardiness increase with increasing U: we observe 3278 time units for $U = 0.7$, 7535 for $U = 0.8$ and 23 510 for $U = 0.9$, respectively.

The area of significant improvements is rather broad, clearly showing the robustness of the approach with respect to the setting of parameters α and, even more, β. Of course, extremely small settings for either α or β eliminate the effect of anticipation.

Setting parameter $\alpha = 0.375$ performs well for all tested utilization levels, although the best found setting increases slightly with increasing U. Deterioration of the tardiness objective can be observed only for very high values of α (which basically means ignoring the original tardiness objective). The maximal value for α to warrant an improvement in tardiness increases with U: the higher the utilization, the more congested the shop floor system becomes, and the more appropriate is a focus on idle time minimization.

An appropriate length of the time interval defined by β apparently depends on the degree of disruption caused by the arrival of new jobs. With regard to the problem at hand, β close to 90 time units seems to be suitable for all U considered. This relatively long time span corresponds to approximately seven to 10 job arrivals (depending on U). The fact that such a long-term consideration of idle times is beneficial indicates that re-scheduling only partially overturns the old schedule.

The right-hand side of figure 3 shows the effect of the parameter variation on the WIP, expressed by the average number of operations observed in a sub-problem. Obviously, a large U imposes a small inter-arrival-time which in turn leads to a large number of operations per sub-problem. Generally, an increasing α decreases the problem size, because the idle time penalty leads to a more effective utilization of machine capacity. This, in turn, results in shorter job flow times and finally in a smaller number of operations per sub-problem. The plots on the right-hand side of figure 3 impressively show the impact of the idle time penalty on WIP. This effect seems more or less independent from the setting of β, as long as it is above some threshold. Obviously, choosing the β value too small renders the α value basically meaningless.

However, although the WIP continues to shrink with increasing α (assuming reasonable β values), beyond a certain α value the tardiness starts to

deteriorate again. This shows that the observed improvements are not due to smaller problem sizes, but rather that the reduction in problem size is a positive side-effect of the focus on flexibility.

Summarizing, anticipatory scheduling is an easily implemented method, robust against parameter settings and yielding remarkable improvements in solution quality.

7. Influence of the weighting scheme

By weighting idle times linearly decreasing with the time of occurrence, we have taken a rather simple and pragmatic approach to emphasize near-term events. It is a valid question whether there exists a better weighting scheme, and what the influence of such a weighting scheme might be. In section 6, we have already examined the influence of the time horizon β and have shown that the results are quite insensitive to this parameter. Snoek (2001) tried to evolve suitable weighting schemes by means of an evolutionary algorithm. His results remain inconclusive, however, mainly due to the large statistical variation and the massive computing power required for such experiments. Even with an iterative refinement procedure, improvements over the linear weighting scheme are non-significant. According to Snoek, the approach seems insensitive to the form of the penalty function, as long as the function is non-increasing with time.

In this section, we report on further experiments using two additional weighting schemes in order to shed some more light on this issue in combination with our proposed approach. First, a uniform weighting of idle times up to the time horizon β is examined. Later on, we investigate an exponentially decreasing weighting.

7.1 *Uniform weighting*

In order to show the importance of a decreasing weighting scheme, we run the same set of experiments as in section 6, but with a uniform weighting of idle times up to time β. Compared to the linearly decreasing weight used so far, we would now expect a much higher influence of the parameter β, because the transition for potential idle time from being fully considered to being not considered at all is now instantaneous rather than gradual. This expectation is confirmed in our experiments.

The upper plots of figure 4 show the improvements in tardiness and WIP per sub-problem, respectively, obtained with uniform weighting for $U = 0.8$. The results for $U = 0.7$ and $U = 0.9$ look correspondingly. Generally, one can observe a moderate decrease in peak performance. Much more important, however, is that the region of parameter settings yielding reasonable performance is much smaller than has been observed for a linearly decreasing weighting scheme. This means that the solution quality becomes very sensitive to parameter β, and also α has to be tuned more carefully. The potential danger of choosing β too large is also evident from the observed WIP: unlike in the linear weighting scheme, the observed number of operations per sub-problem is quite sensitive to the setting of β.

We conclude that although a long-term consideration of machine idle times is desirable, its weight should decrease with increasing time.

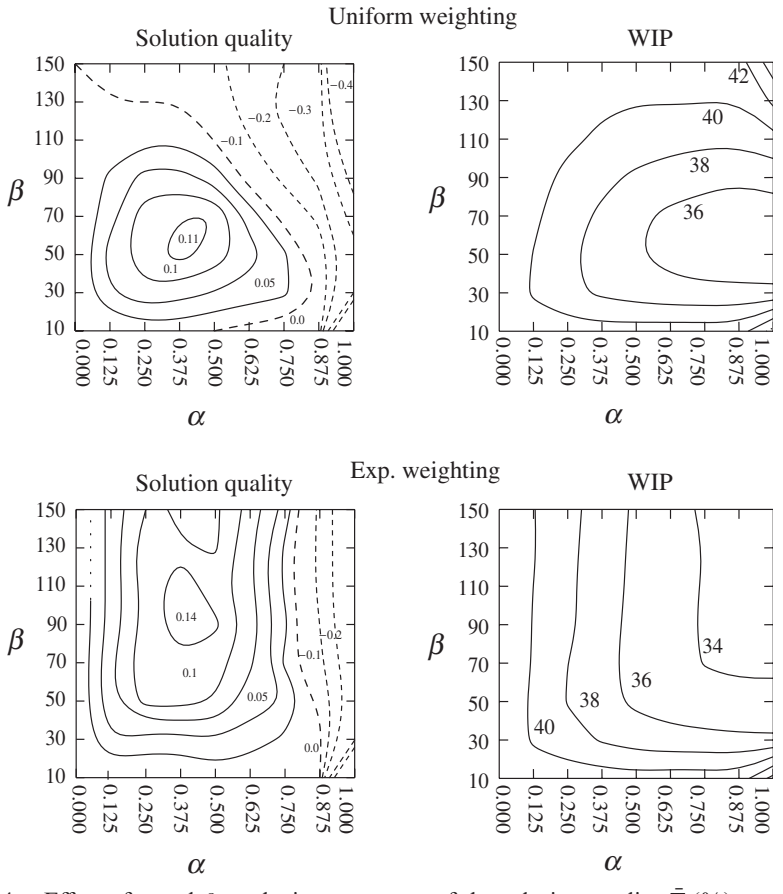

Figure 4. Effect of α and β on the improvement of the solution quality $\bar{T}$ (%) over the result obtained for $\alpha = 0.0$ (left), and the work in process (WIP) depicted as the average number of operations involved in deterministic sub-problems (right). Results obtained from uniform (top) resp. exponential (bottom) weighting of idle times for $U = 0.8$.

7.2 *Exponential weighting*

Since the arrival times in our test environment are exponentially distributed, one might expect that an exponentially decreasing weight might be particularly suitable, reflecting the probability of an idle time of being actually implemented before the arrival of the next job. Therefore, we run another set of experiments with exponential weight distribution. Since the considered time horizon in that case is infinite, β here represents a scaling parameter such that the total area under the weighting function is the same as with the linear weighting, i.e. $w(t) = e^{-2t/\beta}$.

As can be seen in the lower plots of figure 4, the solution quality is more or less equivalent to the quality obtained with linear weighting for basically all values of α and β. One possible explanation might be that rescheduling after the arrival of a new job will not overturn the complete schedule but only modify it slightly. Many idle times of the old schedule will prevail in the new schedule and will eventually be implemented before the arrival of a further job. In that case, an ideal weighting

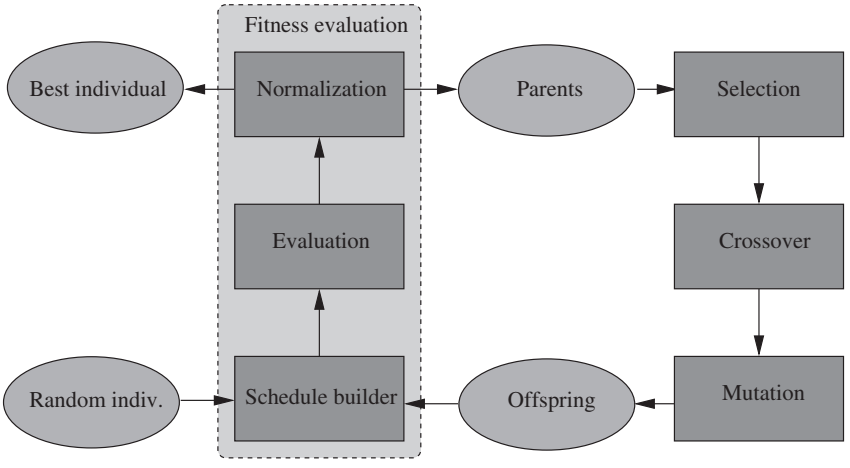

Figure 5. Outline of a generic EA model.

would depend on the expected number of new jobs arriving before the occurrence of a certain idle time, which suggests linear weighting. The effect of β on WIP (right part of figure 4) looks very similar to the linear weighting scheme, which strengthens the conjecture that the specifics of the weighting scheme is not important, as long as it decreases monotonically.

From the above experiments, we conclude that the suggested approach is quite independent of the choice of the weighting function, and that a simple linear weighting scheme suffices. The results also confirm our intuitive reasoning in section 4, namely that early idle times should be penalized more heavily than later idle times.

8. An evolutionary algorithm for dynamic scheduling

BRS is not the only method suitable for combination with our proposed idle time penalty. In fact, a more powerful and particularly suitable class of algorithms seems to be evolutionary algorithms (EAs). Since they maintain a population of solutions throughout the run, they can provide the diversity we need for normalization. And since they work on arbitrary objective functions and do not require any additional data like gradient information or bounds, integrating an idle time penalty is straightforward.

EAs are iterative stochastic search methods based on the principles of natural evolution (Goldberg 1989, Davis 1991, Michalewicz 1996). They have proven to be very powerful and have been successfully applied to many real-world optimization problems, including scheduling (Cheng *et al.* 1996, 1999, Ponnambalam *et al.* 2001).

The outline of our EA is depicted in figure 5. Individuals, representing encoded solutions to the problem under consideration, are produced at random. The fitness of the individuals is determined in three steps. First, the schedule is constructed, then the tardiness and idle time distribution are determined, and finally these measures are normalized resulting in the individual's fitness in the parent population. The best

individual observed so far is stored. Individuals are selected on the basis of their fitness to undergo crossover and mutation operators. The offspring population is evaluated and integrated with the old population to form the new population. This cycle, which is usually called 'generation' in the field of EAs, is performed until a stopping condition is met. In the following, we outline features of the EA used for the experiments in the remainder of this paper.

8.1 *Solution encoding and schedule building*

In order to determine the fitness of an encoded solution, a schedule builder constructs a schedule along the time axis. Thereby, conflicts among competing operations have to be resolved. We encode these decisions by defining priorities between any two operations involved in a problem. These priorities are stored in a permutation consisting of all operations. Whenever two or more operations compete for a machine, the one is given priority which occurs leftmost in the permutation. Basically, a permutation is used analogous to a complex priority rule.

There are two basic forms of schedule builders: those that never leave a machine idle when there is an operation that could be processed (the resulting schedules are called *non-delay* schedules) and those that allow a machine to wait for an urgent operation, although there is another operation that could be processed immediately. This may be advantageous, e.g. if waiting for an almost tardy job can prevent this job from becoming tardy by introducing additional machine idle time.

The maximal reasonable time to wait is given by the smallest processing time among those operations already queued. If one would wait even longer, this least time-consuming operation could be processed in the mean time, which would obviously be a waste of machine capacity. Schedules adhering to this constraint, which means that no single operation can be left-shifted without deteriorating the objective function value, are called *active* (French 1982).

More specifically, a schedule builder operates by iteratively considering machine M' with the earliest possible starting time t' of an operation. For *non-delay* schedules, one of the operations queued in front of M' is picked for dispatching, which can start at $t = t'$, i.e. a machine is never kept idle when there is an operation that might be started. To produce *active* schedules, an operation is determined on M' with a minimal possible completion time c''. Then, an operation is dispatched on M', which can start in the interval $t' \leq t < c''$. Furthermore, we can think of *hybrid* schedules, simply by considering the interval $t' \leq t < t' + (c'' - t')\delta$ with $\delta \in [0, 1]$ defining a bound on the time span a machine is allowed to remain idle (cf. figure 6; see also Storer *et al.* 1992).

From the above considerations, it is obvious that the set of non-delay schedules is a subset of the set of active schedules and that we can scale the size of the set of hybrid schedules by means of δ. It is known from literature that for regular measures of performance like $\bar{T}$, there is at least one optimal schedule in the set of active ones,

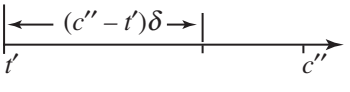

Figure 6. Idle time allowed by setting $\delta = 2/3$.

whereas an optimal non-delay schedule does not necessarily exist. Since active schedules allow for additional machine idle time, there is evidence that non-delay schedules perform much better on average, cf. Storer *et al*. (1992), Della Croce *et al*. (1995), Norman and Bean (1997) and Mattfeld (1999).

In a recent paper, Bierwirth and Mattfeld (1999) showed that using hybrid scheduling in an EA can be successful. In general, it has been shown that the more complex the problem, the smaller the value of δ should be. With a smaller value for δ, the EA profits from searching a smaller search space of a superior mean solution quality. If, on the other hand, the EA potentials are not exhausted, improvements can be gained by extending the search space. The effect of hybrid schedule builders receives particular attention in section 9.

8.2 *Genetic reproduction*

Since sequencing as well as assignment problems allow a permutation encoding, various permutation crossover operators have been developed (Whitley 2000). For vehicle routing, the preservation of precedence relations is assumed to be favourable (Blanton and Wainwright 1993). Bierwirth *et al*. (1996) have incorporated this idea for scheduling problems resulting in the precedence preservative crossover (PPX).

PPX starts with two parent permutations of operations involved in the problem and an empty offspring permutation to be constructed. For the course of the procedure, the permutations are considered as lists. The operator repeatedly selects the leftmost operation randomly from either the first or the second parent and appends it to the offspring. Then, this operation is deleted from both parents. Eventually, we end up with two empty parent permutations and a newly built offspring permutation. Thereby, the PPX operator passes on priorities among operations given in two parental permutations to one offspring at the same rate, while no new priorities are introduced.

The mutation operator alters a permutation by first picking (and deleting) an operation before reinserting this operation at a randomly chosen position of the permutation. At the extreme, the priority relation of one operation to all other operations is affected, but typically a mutation has a much smaller effect.

Besides the above-described components, common EA parameter settings are used in this research. We use a generational reproduction model with a fixed population size of 100 individuals. Selection is based on inverse proportional fitness, the recombination probability is 0.6, and the mutation probability is set to 0.1. We engage a replacement of individuals by an elitist strategy which ensures that the overall superior solution is always passed on to the parent pool of the next generation. For every deterministic sub-problem, similar to BRS we perform 50 000 evaluations by running the EA for 500 generations. Consequently, the overall running time of the EA is similar to that of BRS, namely 3.4 s on a 1.2 GHz PC for a problem with 40 operations.

8.3 *Normalization of objectives*

As has been discussed in section 4, we suggest normalizing the tardiness and flexibility terms by making use of minima and maxima currently observed.

The normalized terms are then combined into a single objective. As the EA works on a population of individuals, that normalization can simply be done on the basis of the current population. As the EA's population changes over time, the normalization adapts accordingly.

8.4 *Using a rolling time horizon*

When scheduling on the basis of a rolling time horizon, two consecutive deterministic problems resemble each other to a certain extent, because typically a few jobs are introduced, and only some operations are discarded. Thus, most of the priority information evolved for a deterministic sub-problem during a EA run will be successfully applicable also in the successive EA for the next sub-problem. This suggests transferring encoded priority information between successive sub-problems, an approach which has been successfully exploited by Lin *et al*. (1997) and Bierwirth and Mattfeld (1999). Thereby, for each individual in the population, old operations are removed from the permutation, and new operations are added at random places.

Thus, priorities among operations to be re-scheduled are passed on to the successive EA. Biasing the population will lead to a faster convergence, which in turn will result in a considerable saving of runtime. Whenever the time available for heuristic search is very limited, the bias of the search towards a promising region of the search space can sometimes even lead to an improved solution quality. To this end, we have taken up this approach also for the experiments in this paper.

9. Evolutionary algorithm and the effect of the schedule builder

Two fundamental concepts for schedule builders have been considered in section 8.1: those that generate active schedules and those that generate non-delay schedules. Although optimality conditions with respect to regular measures of performance suggest searching the set of active schedules, heuristics may benefit from a confinement towards non-delay scheduling in particular in the face of very complex problem instances: the set of non-delay schedules is a subset of the set of active schedules, and a smaller search space is easier to search for a heuristic search method. Moreover, non-delay schedules generally show a better mean performance compared to active schedules, so a search will start from a better level of solution quality.

The concept of anticipatory scheduling suggests an additional viewpoint: non-delay schedule builders implicitly avoid machine idle times whenever possible. Moreover, early idle times are specifically avoided at the expense of later idle time insertions occurring due to the existence of precedence constraints. Hence, for dynamic scheduling a non-delay schedule builder might be particularly advantageous because it implicitly follows the proposed idea of anticipatory scheduling. It might even be possible that non-delay scheduling renders an explicit consideration of flexibility in the objective function unnecessary.

To examine the interdependencies between the schedule builder and the anticipation term, in this section we vary the schedule builder's tendency to build active

Table 2. Percentage of $\bar{T}$ improvement achieved by EA over standard BRS ($\alpha = 0$), depending on α, δ, and U.

		α								
U	δ	0.000	0.125	0.250	0.375	0.500	0.625	0.750	0.875	1.000
0.7	0.00	0.04	0.12	0.15	0.15	0.06	−0.32	−0.39	−0.46	−0.78
	0.25	0.07	0.18	0.20	0.20	0.09	−0.30	−0.39	−0.49	−0.81
	0.50	0.08	0.20	0.22	0.23	0.13	−0.23	−0.33	−0.44	−0.78
	0.75	0.02	0.20	0.22	0.23	0.15	−0.14	−0.27	−0.41	−0.83
	1.00	−0.01	0.19	0.20	0.22	0.17	−0.05	−0.19	−0.33	−0.82
0.8	0.00	0.02	0.08	0.10	0.12	0.12	−0.04	−0.11	−0.16	−0.44
	0.25	0.03	0.10	0.11	0.13	0.13	−0.04	−0.10	−0.17	−0.47
	0.50	0.02	0.10	0.13	0.15	0.16	0.01	−0.07	−0.15	−0.45
	0.75	−0.03	0.10	0.13	0.14	0.15	0.04	−0.03	−0.13	−0.47
	1.00	−0.04	0.06	0.12	0.14	0.15	0.08	0.00	−0.10	−0.48
0.9	0.00	0.06	0.09	0.13	0.15	0.16	0.13	0.09	0.06	−0.22
	0.25	0.05	0.10	0.12	0.15	0.18	0.13	0.09	0.04	−0.22
	0.50	0.04	0.09	0.13	0.16	0.18	0.14	0.09	0.03	−0.22
	0.75	0.00	0.08	0.13	0.14	0.17	0.14	0.09	0.04	−0.24
	1.00	−0.02	0.07	0.12	0.13	0.15	0.14	0.09	0.03	−0.24

Note: The composite fitness function uses a *weighted* idle time and $\beta = 90$.

schedules by means of δ. Experiments are carried out for $U \in \{0.7, 0.8, 0.9\}$ with all combinations of $\alpha \in \{0.000, 0.125, \ldots, 1.000\}$ and $\delta \in \{0.00, 0.25, \ldots, 1.00\}$. Parameter β is set to 90, which has already been identified as appropriate for all U considered. Table 2 lists the percentage of improvement in solution quality ($\bar{T}$) over BRS parameterized with $\alpha = 0.000$ (no anticipation) and non-delay scheduling.

Let us first focus on the performance of the EA when used in combination with standard non-delay scheduling (i.e. the three rows with $\delta = 0$). Without anticipation ($\alpha = 0$), we observe slightly better results than with BRS (by 4, 2, and 6% for $U = 0.7$, 0.8, and 0.9, respectively). This indicates that the EA used is, by itself, a competitive approach. As expected, the introduction of the flexibility term by increasing α improves the performance significantly, by 11 percentage points for $U = 0.7$,[2] 10 percentage points for $U = 0.8$[3] and 10 percentage points for $U = 0.9$[4]. This is in a similar range as the 14–15% improvement observed for BRS (cf. section 5), which indicates that the benefit of anticipation is more or less independent of the underlying optimization or heuristic search method. However, in combination with the EA, the benefit is much more sensitive to the setting of parameter α. If α is set too large, performance may deteriorate by as much as 83%.[5] This is not surprising, as strongly focusing on the avoidance of early idle times and ignoring completely the original tardiness objective ($\alpha = 1.0$) may lead to arbitrarily bad tardiness values. BRS is not so sensitive, because by using the priority rules, BRS produces only

[2]Compare ($\alpha = 0.0, \delta = 0.0$) with ($\alpha = 0.25, \delta = 0.0$).
[3]Compare ($\alpha = 0.0, \delta = 0.0$) with ($\alpha = 0.375, \delta = 0.0$).
[4]Compare ($\alpha = 0.0, \delta = 0.0$) with ($\alpha = 0.5, \delta = 0.0$).
[5]See $U = 0.7,\ \delta = 0.75,\ \alpha = 1.0$.

reasonable schedules with respect to the tardiness objective, and the flexibility criterion is only applied to choose among these.

Now let us examine the effect of the schedule builder on a standard EA without anticipation (i.e. the column with $\alpha = 0$). Increasing δ and thus moving from a non-delay scheduler towards active schedulers allows the solution quality to be improved slightly, up to 8, 3, and 6% over BRS depending on utilization. The overall effect of δ, though, is surprisingly small and for very large values of δ, results even deteriorate by up to 4% ($\delta = 1$ and $U = 0.8$). The best setting for δ is 0.5 for $U = 0.7$, 0.25 for $U = 0.8$ and 0.0 for $U = 0.9$, i.e. the higher the utilization, the lower δ should be. This corresponds to the findings in Bierwirth and Mattfeld (1999) and may be attributed to the following two reasons: first, sub-problems for large U are more difficult to solve, and therefore the reduced problem complexity resulting from a non-delay schedule builder outweighs the potential improvements that might be obtained by using an active schedule builder. Second, a non-delay schedule builder implicitly avoids early machine idle times and thus, as an (originally unintended) side-effect, may increase flexibility. That flexibility is more important in highly utilized shop floors.

Looking at the remaining entries of the table, there is another interesting aspect to be noted: when the idle time penalty is used sensibly ($0.25 \leq \alpha \leq 0.5$), the EA may exploit the opportunities of active scheduling much better than without an idle time penalty, and increasing δ is much more effective than when anticipation is not used ($\alpha = 0$). More specifically, for $\alpha = 0$, tuning δ improves the solution quality by up 4 percentage points (depending on the utilization level), while for $\alpha = 0.375$, the improvement which can be obtained by tuning δ can be as large as 8 percentage points. Additionally, the best found setting for δ is higher when an idle time penalty is used than when it is not used. These observations support the hypothesis that non-delay scheduling has a similar effect as the idle time penalty (by avoiding early idle times) but is at best a much cruder form of anticipation, as it cannot distinguish between useful and useless idle time (as when using an idle time penalty, where the benefit of including idle time is explicitly considered against the penalty).

As has already been observed with the combination of idle time penalty and BRS in section 6, the higher the shop-floor utilization, the higher should be the value for α. Nevertheless, the parameter settings are quite robust, and $\alpha = 0.375$ in combination with $\delta = 0.5$ seems to work well independent of U.

Overall, the EA may achieve impressive improvements of 23%, 16%, and 18% (depending on U) over BRS without anticipation! Most of this improvement can be attributed to the use of an idle time penalty; only some of it is due to using an EA and tuning δ. This again demonstrates the effectiveness of our approach. Non-delay scheduling, although it seems to have a similar effect in some sense, is certainly no substitute for an early idle time penalty.

10. Comparison with priority rules

The primary goal of this paper is to provide evidence for our hypothesis that better schedules can be obtained by anticipatory scheduling, rather than to construct the absolute best scheduling algorithm. Nevertheless, it is certainly interesting to see how BRS and the EA used for testing perform when compared to other scheduling

Table 3. Improvement of different heuristics relative to standard BRS ($\alpha = 0$), in percent ±S.E.

	U		
Heuristic	0.7	0.8	0.9
BRS ($\alpha = 0$)	0	0	0
SPT	-37 ± 1.1	-26 ± 1.0	-16 ± 1.3
RR rule	-26 ± 1.7	-17 ± 0.7	-7 ± 1.1
EA ($\alpha = 0, \delta = 0.5$)	8 ± 1.3	2 ± 0.9*	4 ± 0.6
BRS ($\alpha = 0.375, \beta = 90$)	15 ± 1.0	12 ± 1.3	14 ± 0.8
EA ($\alpha = 0.375, \beta = 90, \delta = 0.5$)	23 ± 0.8	15 ± 0.9	16 ± 1.0

Note: Except for the case marked '*', all results differ significantly from the standard BRS result according to a two-sided t-test with level of significance = 99%.

approaches. Since solving the problems to optimality is computationally very demanding, we rely on standard heuristics for comparison.[6]

Table 3 compares a number of approaches to our usual baseline, namely BRS without anticipation ($\alpha = 0$). As can be seen, BRS outperforms the simple yet usually quite effective (at least for problems with high utilization) shortest processing time (SPT) priority rule by a large margin. This result is to be expected, since SPT is used inside BRS. The difference decreases with increasing utilization, as SPT is a particularly good rule in highly congested shop floors.

In a recent comparative study by Rajendran and Holthaus (1999), the best priority rule for dynamic total tardiness job-shop scheduling problems was found to be the RR rule, which we therefore consider state of the art. This rule has been suggested by Raghu and Rajendran (1993) and takes into account the probable waiting time of the job at the machine of the job's next operation. More precisely, the priority of operation j of job i is calculated as follows:

$$\mathrm{RR} = \left(s_i \cdot e^{-U} \cdot p_{ij}\right)/\mathrm{RPT}_i + e^{U} \cdot p_{ij} + W_{next},$$

where s_i is the slack of job i, p_{ij} is the operation's processing time, RPT_i denotes the sum of process times of uncompleted operations, W_{next} indicates the probable waiting time of job i at the machine of its next operation taking into account all operations currently in the queue, and U denotes the utilization level. Note that in our tests, we gave the RR rule a slight advantage by providing it with the utilization level used to generate the (balanced) problem instances, something that may be rather difficult to do for real-world problems.

As the results show, RR clearly outperforms SPT, which is in accordance with the findings by Rajendran and Holthaus (1999). Nevertheless, BRS significantly outperforms RR on all utilization levels. The EA is still slightly better than BRS, in particular on the smaller problems with low utilization. This demonstrates that BRS and the EA used are indeed adequate for the problem at hand, although they

[6]Furthermore, since the problem has to be solved on a rolling time horizon with jobs being revealed only over time, even solving each sub-problem to optimality would most likely not produce globally optimal results.

of course require significantly more time (they generate around 50 000 schedules for each sub-problem, compared to only one for SPT and RR).

When combining BRS and EA with our suggested penalty term, the performances of both methods improve by another 12–15 percentage points.

11. Nervousness considerations

In many practical applications, raw material is ordered, and promises are made to customers well in advance, based on the planned schedule. When the schedule is changed to accommodate additional jobs, such changes may then create conflicts with suppliers and customers.

Therefore, such changes should usually be kept as small as possible. The extent by which the re-scheduling algorithm causes such changes is also called 'nervousness' and can be an important criterion to evaluate a re-scheduling algorithm. Therefore, in this section, we will report on the nervousness generated by the different scheduling algorithms.

Note that we do not explicitly attempt to minimize nervousness, but simply observe the nervousness resulting from the different re-scheduling paradigms considered in this paper. For a recent paper on how to explicitly integrate re-scheduling cost into the optimizer, see Hall and Potts (2004), where re-scheduling is considered for different goals and nervousness measures.

As a measure of nervousness, we use the sum of the differences in the start times of operations, from the backlog of one problem to the schedule of the next, over all scheduling problems generated within one run. A similar measure has been used, e.g. by Wu *et al.* (1992).

Table 4 compares the nervousness resulting from the primary scheduling paradigms considered in this paper, again relative to our baseline algorithm BRS.

As can be seen, the greatest reduction in nervousness compared to BRS is achieved by SPT. That is not surprising, since SPT is a very simple rule, and the priority of an operation only depends on its processing time. In other words, an additional job will be inserted somewhere and move some other operations backwards, but otherwise the sequence of jobs will be largely unchanged.

Table 4. Nervousness improvement of different heuristics relative to standard BRS ($\alpha = 0$), in percent ±S.E.

	U		
Heuristic	0.7	0.8	0.9
BRS ($\alpha = 0$)	0	0	0
SPT	36 ± 1.5	50 ± 1.6	70 ± 1.4
RR rule	27 ± 1.7	38 ± 1.1	53 ± 1.6
EA ($\alpha = 0, \delta = 0.5$)	29 ± 2.2	44 ± 1.4	58 ± 1.2
BRS ($\alpha = 0.375$, $\beta = 90$)	9 ± 1.5	19 ± 1.4	29 ± 1.0
EA ($\alpha = 0.375$, $\beta = 90, \delta = 0.5$)	36 ± 2.3	48 ± 1.3	62 ± 1.3

Note: All results differ significantly from the standard BRS result according to a two-sided t-test with level of significance = 99%.

On the other hand, BRS may completely overturn an old schedule if this helps to reduce tardiness.

RR is also a relatively fixed rule and thus reduces nervousness significantly compared to BRS. However, since RR implements some form of look ahead by taking into account the situation at the next machine, it is more likely that additional jobs lead to changes in the schedule even before the start time of the job, and thus the reduction in nervousness is smaller than with SPT.

The EA exhibits a significantly smaller nervousness than BRS, which is probably a side-effect of seeding each run with the best solution of the previous run, thereby biasing the search towards the neighbourhood of the old solution.

Introducing our proposed flexibility measure significantly reduces nervousness for BRS (compare rows BRS ($\alpha = 0$) and BRS ($\alpha = 0.375, \beta = 90$)) as well as the EA (compare rows EA ($\alpha = 0, \delta = 0.5$) and EA ($\alpha = 0.375, \beta = 90, \delta = 0.5$)), although the effect is stronger for BRS than EA. Thus, the consideration of nervousness can be seen as an additional support for our proposed anticipative scheduling approach.

Overall, looking at tardiness (table 3) and nervousness (table 4), there are only three approaches that dominate BRS (i.e. that outperform BRS in both criteria), namely the standard EA, anticipatory BRS and anticipatory EA. Overall, only two approaches are efficient (not dominated by any other method): the anticipatory EA, which reduces tardiness by 15% and nervousness by 48%, and SPT, which reduces tardiness slightly more (50%) but increases tardiness by 26% and which is usually not acceptable.

12. Problem variants

In order to allow our extensive tests to be run in reasonable time, the test instances used so far have been rather small. In this section, we want to examine how well anticipative scheduling works with a less rigid due-date configuration and also for larger problems. While in the original problem set, the due date of an operation i was always set as $d_i = r_i + 2 \cdot p_i$, in setting B, the deadlines are randomly generated from $[r_i + p_i, r_i + 3p_i]$. In setting C, we use 12 instead of six machines. Correspondingly, the number of operations per job is changed from between four and six to between eight and 12. All other settings remain the same as described in section 3, and results are reported for $U = 0.8$. Because the fine-tuning of parameters to the problem instance is usually not possible in the real world, we simply used the same parameters for the rescheduling algorithms that had proven successful on the original problem, i.e. a time horizon of $\beta = 90$ and a weight $\alpha = 0.375$.

The results of the different algorithms are depicted in table 5. When comparing these results with table 3, it can be seen that the distribution of due dates has only a small effect on performance, not changing the relative performance of the considered heuristics. The relative ranking of the heuristics remains also the same for the larger problem instances, but we can see two effects: for once, the EA produces much better results than BRS, presumably because with more machines and operations, there is a larger potential in optimization that can be harvested by the EA. Second, anticipation becomes even more important, with improvements of around 30% (compared to 12% for the small problem). Overall, the results also demonstrate the robustness

Table 5. Performance of different heuristics relative to standard BRS ($\alpha = 0$), improvement in percent ±S.E.

	Problem setting	
Heuristic	B	C
BRS ($\alpha = 0$)	0	0
SPT	-32.2 ± 1.1	-24.5 ± 1.7
RR rule	-20.6 ± 1.3	-9.5 ± 1.5
EA ($\alpha = 0, \delta = 0.5$)	1.6 ± 3.2	21.3 ± 1.3
BRS ($\alpha = 0.375, \beta = 90$)	10.4 ± 0.7	31.4 ± 1.2
EA ($\alpha = 0.375, \beta = 90, \delta = 0.5$)	12.1 ± 4.4	35.1 ± 1.5

of our approach with respect to problem instance and parameter setting, as we had simply re-used settings from previous tests.

13. Conclusion

Many real-world problems are dynamic and change over time, requiring repeated re-optimization as new information becomes available. A standard approach to deal with these dynamics is to use a rolling time horizon. The problem is decomposed into a series of static sub-problems, which are then solved independently.

In this paper, we have argued that, if the decisions at one stage influence the problems encountered in the future, future changes need to be anticipated by searching not only for good solutions but for solutions that additionally influence the state of the problem in a favourable way. We have identified these solutions as being *flexible*, i.e. easily adjustable to changes in the environment.

To validate this conjecture, we have considered a dynamic stochastic total tardiness job-shop scheduling problem, with jobs arriving non-deterministically over time. For this particular application, we have conjectured that the avoidance of early idle times can support the flexibility of a solution.

As a consequence, we have suggested extending the objective function to additionally penalize early idle times. This idea was tested in combination with a randomized priority rule scheduler as well as an evolutionary algorithm. It showed impressive improvements in solution quality in comparison to a corresponding algorithm without flexibility term. Obviously, short-term efficiency losses due to a flexibility term added to the original objective function lead to significant long-term gains of efficiency in a dynamic environment.

There are several possibilities for future research. First of all, although penalizing early idle times clearly proved to fulfill the goal of increasing flexibility, it may not be the only helpful criterion for flexibility. Another one may be, for example, to minimize dependencies in the later part of the schedule. Since we basically work with two objectives, tardiness and idle time penalty, a multi-objective evolutionary algorithm may help to maintain diversity and improve results over the single-objective evolutionary algorithm.

The basic approach presented in this paper is not necessarily restricted to job-shop scheduling. In fact, we expect that it would prove beneficial for other

dynamic stochastic problems where optimization decisions influence the future system state, as for example in transportation, where new customer requests have to be integrated into the vehicles' routes. Of course, appropriate flexibility measures have to be developed which capture the critical aspects. Transferring the presented framework to other problem domains will be subject to future work.

Acknowledgements

We would like to thank the anonymous referees for their valuable suggestions and comments.

References

Adam, N. and Surkis, J., Priority update intervals and anomalies in dynamic ratio type job shop scheduling rules. *Manage. Sci.*, 1980, **26**, 1227–1237.

Baker, K., *Introduction to Sequencing and Scheduling*, 1974 (Wiley: New York).

Baker, K.R., An experimental study of the effectiveness of rolling schedules in production planning, *Decis. Sci.*, 1977, **8**, 19–27.

Bierwirth, C. and Mattfeld, D.C., Production scheduling and rescheduling with genetic algorithms. *Evolut. Comput.*, 1999, **7**(1), 1–18.

Bierwirth, C., Mattfeld, D.C. and Kopfer, H., On permutation representations for scheduling problems. In *Parallel Problem Solving from Nature IV*, edited by H.-M. Voigt, W. Ebeling, I. Rechenberg and H.-P. Schwefel, pp. 310–318, 1996, (Springer: New York).

Blanton, J.L. and Wainwright, R.L., Multiple vehicle routing with time and capacity constraints using genetic algorithms. In *Proceedings of the 5th International Conference on Genetic Algorithms*, edited by S. Forrest, pp. 452–459, 1993 (Morgan Kaufmann: San Mateo, CA).

Branke, J., *Evolutionary Optimization in Dynamic Envionments*, 2001 (Kluwer: Dordrecht).

Branke, J. and Mattfeld, D., Anticipation in dynamic optimization: the scheduling case. In *Parallel Problem Solving from Nature (PPSN VI)*, edited by M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J.J. Merelo and H.-P. Schwefel, Volume 1917 of *LNCS*, pp. 253–262, 2000 (Springer: New York).

Chang, A.-Y., Whitehouse, D.J., Chang, S.-L. and Hsieh, Y.-C., An approach to the measurement of single-machine flexibility. *Int. J. Prod. Res.*, 2001, **39**(8), 1589–1601.

Cheng, R., Gen, M. and Tsujimura, Y., A tutorial survey of job-shop scheduling problems using genetic algorithms. *Comput. Ind. Eng.*, 1996, **30**, 983–997.

Cheng, R., Gen, M. and Tsujimura, Y., A tutorial survey of job-shop scheduling problem using genetic algorithms—II: hybrid genetic search strategies. *Comput. Ind. Eng.*, 1999, **36**, 343–364.

Church, L. and Uzsoy, R., Analysis of periodic and event-driven rescheduling policies in dynamic shops. *Int. J. Comput. Integ. Manuf.*, 1992, **5**, 153–163.

Davis, L., *Handbook of Genetic Algorithms*, 1991 (Van Nostrand Reinhold: New York).

Della Croce, F., Tadei, R. and Volta, G., A genetic algorithm for the job shop problem. *Comput. Operat. Res.*, 1995, **22**, 15–24.

Drexl, A., Scheduling of project networks by job assignment. *Manage. Sci.*, 1991, **37**, 1590–1602.

Fang, J. and Xi, Y., A rolling horizon job shop rescheduling strategy in the dynamic environment. *Int. J. Adv. Manuf. Technol.*, 1997, **13**(3), 227–232.

Farn, C.-K. and Muhlemann, A., The dynamic aspects of a production scheduling problem. *Int. J. Prod. Res.*, 1979, **17**, 15–21.

French, S., *Sequencing and Scheduling; An Introduction to the Mathematics of the Job-shop*, 1982 (Ellis Horwood: New York).

Goldberg, D.E., *Genetic Algorithms*, 1989 (Addison-Wesley: Reading, MA).
Gupta, Y.P. and Somers, T.M., The measurement of manufacturing flexibility. *Eur. J. Operat. Res.*, 1992, **60**, 166–182.
Hall, N.G. and Potts, C.N., Rescheduling for new orders. *Operat. Res.*, 2004, **52**(3), 440–453.
Haupt, R., A survey of priority rule-based scheduling. *Operat. Res. Spek.*, 1989, **11**, 3–16.
Holthaus, O. and Rajendran, C., Efficient dispatching rules for scheduling in a job shop. *Int. J. Prod. Econ.*, 1997, **48**, 87–105.
Jain, A.K. and Elmaraghy, H.A., Production scheduling/rescheduling in flexible manufacturing. *Int. J. Prod. Res.*, 1997, **35**(1), 281–309.
Jensen, M.T., Improving robustness and flexibility of tardiness and total flow time job shops using robustness measures. *Appl. Soft Comput.*, 2001, **4**, 1–18.
Kimenmia, J. and Gershwin, S.B., An algorithm for the computer control of a flexible manufacturing system. *IIE Trans.*, 1983, **15**, 353–362.
Kutanoglu, E. and Wu, S.D., *Improving Schedule Robustness via Stochastic Analysis and Dynamic Adaptation*. Technical Report 98T-001, Lehigh University, 1998.
Leon, V.J., Wu, S.D. and Storer, R.H., Robustness measures and robust scheduling for job shops. *IIE Trans.*, 1994, **26**(5), 32–43.
Li, R.-K., Shyu, Y.-T. and Adiga, S., A heuristic rescheduling algorithm for computer-based production scheduling systems. *Int. J. Prod. Res.*, 1993, **31**(8), 1815–1826.
Lin, S.-C., Goodman, E.D. and Punch, W.F., A genetic algorithm approach to dynamic job shop scheduling problems. In *International Conference on Genetic Algorithms*, edited by T. Bäck, pp. 481–488, 1997 (Morgan Kaufmann: San Mateo, CA).
Mattfeld, D., Scalable search spaces for scheduling problems. In *Genetic and Evolutionary Computation Conference*, edited by W. Banzaf, J. Daida, A. Eiben, M. Garzon, V. Honavar, M. Jakiela, and R. Smith, pp. 1616–1621, 1999 (Morgan Kaufmann: San Mateo, CA).
Mehta, S. and Uzsoy, R., Predictable scheduling of a job-shop subject to breakdowns. *IEEE Trans. Robot. Automat.*, 1998, **14**(3), 365–378.
Michalewicz, Z., *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd edn, 1996 (Springer: New York).
Morton, T.E. and Pentico, D.W., *Heuristic Scheduling Systems*, 1993 (Wiley: Chichester, UK).
Muhlemann, A., Lockett, A. and Farn, C.-K., Job shop scheduling heuristics and frequencies of scheduling. *Int. J. Prod. Res.*, 1982, **20**, 227–241.
Norman, B. and Bean, J., A random keys genetic algorithm for job shop scheduling problems. *Eng. Design Automat. J.*, 1997, **3**, 145–156.
Pagell, M., Newman, W.R., Hanna, M.D. and Krause, D.R., Uncertainty, flexibility, and buffers: three case studies. *Prod. Invent. Manage. J.*, 2000, **41**(1), 35–43.
Pereira, J. and Paulré, B., Flexibility in manufacturing systems: a relational and a dynamic approach. *Eur. J. Operat. Res.*, 2001, **139**, 70–82.
Persentili, E. and Alptekin, S.E., Product flexibility in selecting manufacturing planning and control strategy. *Int. J. Prod. Res.*, 2000, **38**(9), 2011–2021.
Pierreval, H. and Mebarki, N., Dynamic selection of dispatching rules for manufacturing system scheduling. *Int. J. Prod. Res.*, 1997, **35**(6), 1575–1591.
Ponnambalam, S.G., Aravindan, P. and Rao, P.S., Comparative evaluation of genetic algorithms for job-shop scheduling. *Prod. Plan. Cont.*, 2001, **12**, 560–574.
Powell, W.B., Towns, M.T. and Marar, A., *On the Value of Optimal Myopic Solutions for Dynamic Routing and Scheduling Problems in the Presence of User Non-compliance*. Technical Report SOR-97-15, Princeton University, Princeton, NJ, 1998.
Raghu, T.S. and Rajendran, C., An efficient dynamic dispatching rule for scheduling in a job shop. *Int. J. Prod. Econ.*, 1993, **32**, 301–313.
Rajendran, C. and Holthaus, O., A comparative study of dispatching rules in dynamic flowshops and jobshops. *Eur. J. Operat. Res.*, 1999, **116**, 156–170.
Raman, N. and Talbot, F.B., The job shop tardiness problem: a decomposition approach. *Eur. J. Operat. Res.*, 1993, **69**, 187–199.
Russel, R., Dar-El, E. and Taylor, B., A comparative analysis of the COVERT job sequencing rule using various shop performance measures. *Int. J. Prod. Res.*, 1987, **25**, 1523–1540.

Snoek, M., Anticipation optimization in dynamic job shops. In *GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems*, pp. 43–46, 2001.

Stadler, H., Production planning and scheduling. In *Supply Chain Management and Advanced Planning*, edited by H. Stadler and C. Kilger, 3rd edn, pp. 197–214, 2005 (Springer: New York).

Storer, R., Wu, D. and Vaccari, R., New search spaces for sequencing problems with application to job shop scheduling. *Manuf. Sci.*, 1992, **38**, 1495–1509.

Whitley, D., Permutations. In *Evolutionary Computation 1*, edited by T. Bäck, D. Fogel and T. Michalewicz, pp. 139–150, 2000 (IOP Press: New York).

Wu, S.D., Storer, R.H. and Chang, P.-C., A rescheduling procedure for manufacturing systems under random disruptions. In *New Directions for Operations Research in Manufacturing*, edited by G. Fandel, T. Gulledge and A. Jones, pp. 292–308, 1992 (Springer: New York).

Yamamoto, M. and Nof, S., Scheduling/rescheduling in the manufacturing operating system environment. *Int. J. Prod. Res.*, 1985, **23**, 705–722.

Yellig, E.J. and Mackulak, G.T., Robust deterministic scheduling in stochastic environments: the method of capacity hedge points. *Int. J. Prod. Res.*, 1997, **35**(2), 369–379.