

TECHNICAL REPORT 1999

**Efficient Adaptive Algorithms for
Dynamic Job Shop Scheduling**

Dirk C. Mattfeld and Christian Bierwirth

Univ. of Bremen
Dept. of Economics
Box 330 440
D-28334 Bremen
Germany

Email: dirk@logistik.uni-bremen.de
Tel.: ++49-421-218-2011
Fax.: ++49-421-218-4271

Efficient Adaptive Algorithms for Dynamic Job Shop Scheduling

Dirk C. Mattfeld and Christian Bierwirth
Dept. of Business Studies & Economics
University of Bremen, Germany

Abstract

Many recent algorithmic approaches suffer from their limited applicability to dynamic scheduling scenarios. Genetic Algorithms have been shown to overcome this problem at the expense of searching less efficient. We propose a flexible dispatching procedure which is controlled by a Genetic Algorithm. For this algorithm we investigate the impact of scaling the search space heuristically. Then we apply a temporal decomposition to the dynamic scheduling problem and discuss efficiency issues of the resulting adaptive algorithm. Finally we compare the outcome with priority-rule based dispatching and with other adaptive approaches reported in literature.

Keywords: Dynamic Job Shop Scheduling, Genetic Algorithm, Priority Rules, Tunable Dispatching, Search Space Decomposition

1 Introduction

Within the last two decades an enormous progress in scheduling research is observed. Attracted by the difficulty of *NP*-hardness, academic research has predominantly focused on standard optimisation models. For these models, very efficient heuristics, often based on local-search, have been developed [6]. Their success can be explained by the simplicity of the objective and constraints which allow the formulation of sophisticated neighbourhood definitions. Unfortunately, the efficiency of such neighbourhoods gets easily lost if real-world constraints stemming from specific production requirements and additional job data are involved in the optimisation.

Also research on job shop scheduling has concentrated on a standard model which permits to take advantage from a sophisticated schedule representation known as the disjunctive graph formulation [1]. For this model local-search algorithms like Simulated Annealing and Tabu Search have been applied with great success [2]. Unfortunately this formulation restricts application to makespan minimisation problems. While these developments have not found their way into practise, today's management of scheduling systems still relies on priority rule based dispatching.

Priority rules have been studied intensively already in the early days of scheduling research. Different to recent work, rather complex simulation models have been used to verify the value of a rule under specific conditions. However,

the relation between applying a certain rule and pursuing a certain objective is not always clear. At the extreme, the control of the dispatcher may even contradict the scheduling objective. Therefore it is desirable to integrate the dispatching activity into an optimisation framework which allows to adapt the decision making process.

A flexible framework for adaptive search is provided by the Genetic Algorithm (GA) paradigm. Its basic idea is to search an encoded solution space by performing syntactical operations on a population of candidate solutions. Since this framework is problem independent to a large extent, it has been applied in almost all fields of optimisation. For many problems GAs have proven powerful, although for job shop scheduling GAs turned out less efficient. For real-world scheduling scenarios however, their universal applicability may warrant GAs to become a serious alternative to priority rule based dispatching [19].

In this paper we propose a pretty standard GA utilised for dynamic job shop scheduling. The GA evolves encoded priorities for operations which then control the dispatching activity. We address the issue of GA weakness by proposing two passable ways of cutting down the search space of a scheduling problem. Although a decrease of the search space may exclude optima from being considered, the GA can benefit from searching a smaller space.

A benchmark set of dynamic job shop scenarios is available from [24]. Release dates, due dates and job weights are considered which allows to pursue flow-time as well as tardiness related objectives. Since neither optima nor adequate lower bounds are known for these problems, one may suspect that a GA is the first choice only because of the absence of other rivals. Fortunately the benchmarks have already been tackled by two other approaches [16, 21] which provide a basis for comparison.

In Section 2 we introduce the dynamic job shop model before we present the benchmark set in detail. In Section 3 we discuss priority rules and present results obtained for the benchmark set. Next we describe the GA in Section 4 before we list results obtained with two different dispatching procedures. In Section 5 we describe a tunable dispatcher introduced in [29] and already used within a GA in [8]. This dispatcher is capable of scaling the search space through a continuous parameter and can produce schedules in the range of non-delay to active ones. By means of this instrument we reveal a GA weakness and propose a remedy. In Section 6 we engage a temporal decomposition for dynamic scheduling problems, compare [27], to further reduce the search space. Again we exclude parts of the search space by enhancing the GA capabilities at the same time. In Section 7 we compare the results obtained with other approaches before we conclude.

2 Dynamic Job Shop Scheduling Problems

Practical scheduling problems tend to be of dynamic rather than of static nature. In industrial production jobs are released for being processed with respect to capacity constraints, material supplies and customer demands. Usually a release-date is announced for a job which denotes the earliest possible starting time of processing the first operation of that job. A due-date is assigned to the job indicating a point in time at which a job should be completed in order to keep the projected date of delivery. The planning further considers priorities among the jobs. Such priorities indicate the effort which has to be spent on

completing a job on time with respect to the importance of a certain customer.

In this context, makespan minimisation is no adequate measure of schedule performance, because the makespan crucially depends on the job with the latest release-date. A flow-time dependent criterion can be used instead for schedule evaluation. This makes sense if a moderate work-in-process is pursued for reasons of shop floor flexibility or inventory-holding costs. Furthermore due-date oriented criteria are considered which stems from the desire to improve customer service or to avoid penalties for late deliveries.

2.1 Modelling Dynamic Job Shops

The standard job shop model considers n jobs to be processed on m machines [14]. The processing of a job on a certain machine is referred to as operation. The operation processing times are fixed and known in advance. Every job is processed in a prescribed technological order, called its machine routing. In the standard model every job passes every machine exactly once resulting in a total of $n \times m$ operations. In this research we modify the standard job shop model towards dynamic scheduling scenarios by considering

- non zero release-dates r_i of jobs,
- prescribed job due-dates d_i , and
- priorities among the jobs, expressed by weights $\omega_i > 0$.

Thus, no job can start before its release-date r_i and its processing should not exceed its due-date d_i . In the scenarios considered a job does not necessarily pass every machine, moreover it may pass some machines more than once.

Among a variety of criteria for schedule optimisation, the following objectives are treated in this study

- the minimisation of the weighted mean flow-time of jobs \bar{F} ,
- the minimisation of the weighted mean tardiness of jobs \bar{T} ,
- the minimisation of the maximum tardiness of jobs T_{\max} , and
- the minimisation of the weighted number of tardy jobs T_n .

A definition of the weighted mean flow-time refers to the completion time c_i of jobs, which is determined by the starting time of the final operation of a job plus the associated operation processing time. This figure is used in the well-known measure

$$\bar{F} = \frac{1}{n} \sum_{i=1}^n \omega_i (c_i - r_i). \quad (1)$$

The tardiness of a job indicates the time span the completion time overshoots the projected due-date. Let $T_i = \max(c_i - d_i, 0)$ denote the tardiness of a job. Based on this measure, the weighted mean tardiness is defined as

$$\bar{T} = \frac{1}{n} \sum_{i=1}^n \omega_i T_i. \quad (2)$$

Table 1: A benchmark set of heterogeneous scheduling scenarios.

no.	n	m	ops	A	U	σ_u	σ_w
1	30	3	67	1.4	0.7	0.1	0.5
2	30	3	60	1.5	0.9	0.4	0.6
3	30	3	126	1.1	0.7	0.2	0.5
4	30	6	216	1.6	1.0	0.3	0.5
5	30	6	292	0.8	0.2	0.1	0.5
6	30	6	196	1.0	0.4	0.2	0.5
7	50	5	193	1.5	0.8	0.2	0.3
8	50	5	466	1.0	0.5	0.1	0.3
9	50	5	211	1.5	1.0	0.3	0.5
10	50	8	250	1.5	0.7	0.1	0.6
11	50	8	318	0.0	0.4	0.2	0.3
12	50	8	230	1.5	0.9	0.3	0.4

A related criteria measures the worst violation of the due-dates

$$T_{\max} = \max\{T_i \mid 1 \leq i \leq n\}. \quad (3)$$

To define a further measure, let $tardy(i)$ indicate whether job i is tardy or not, i.e. $tardy(i) = 1$ means job i is tardy ($T_i > 0$) while $tardy(i) = 0$ means it is completed in time. The weighted number of tardy jobs is simply given by

$$T_n = \sum_{i=1}^n \omega_i tardy(i). \quad (4)$$

As it can be recorded easily, this measure is frequently met in practise because it has a direct relation to the logistical β service-level which designates the percentage of on-time deliveries.

2.2 Dynamic Scheduling Scenarios

For a computational study we rely on twelve benchmark problems provided with the ‘‘Parsifal’’ software package in [24]. It can be taken from Table 1 that the problems no. 1-6 consist of thirty jobs each, to be scheduled on three and six machines respectively. The larger problems no. 7-12 consist of fifty jobs and five to eight machines. These problems contain about three times the operations of the smaller ones. The total number of operations ops is within the range of $n \times m$. A smaller value of ops (e.g. problem no. 1) indicates that not every job passes every machine while a larger value of ops (e.g. scenario no. 3) indicates that some jobs pass a machine at least twice.

The time span between the release-date and the due-date of a job is called its allowance. Let P_j denote the total processing time of a job, i.e. the sum of its operation processing times. The allowance can be given in percent of the total processing time

$$A_j = \frac{d_j - r_j}{P_j}, \quad (5)$$

expressing the tightness of a job’s due-date. Obviously, if $A_j < 1$ a job will always be tardy. However, completing a job in time is not very likely as well if A_j is close to 1. An allowance up to 1.5 indicates a rather tight due-date.

Considering job due-dates alone does not sufficiently reflect the complexity of a dynamic scheduling scenario. Also the setting of release-dates influences the problem difficulty. Frequent job releases lead to a high workload, because many jobs concurrently await processing at a certain point in time. Jobs awaiting processing at the same machine are conflicting. The more such conflicts occur, the more interdependent the consecutive dispatching decisions get. This of course has a strong impact on the difficulty of a scheduling problem.

The observed workload is caused by two independent forces, the arrival process of jobs and the scheduling process itself. The former force is described by the mean inter-arrival-time of jobs

$$\lambda = \frac{1}{n-1} \sum_{j=1}^{n-1} (r_{j+1} - r_j). \quad (6)$$

Due to the a priori unknown quality of scheduling, the machine utilisation rate is used to approximate a certain workload situation. Let \bar{P} denote the average total processing time of all jobs belonging to a scenario, i.e. $\bar{P} = \sum_j P_j/n$. Since at most m machines are busy in parallel, the utilisation rate is given by

$$U = \frac{\bar{P}}{m\lambda}. \quad (7)$$

A value $U > 1$ leads to a continuous increase of the workload over time. For a utilisation ranging from 0.7 to 1.0, challenging scenarios can be expected in general.

Another typical issue of scheduling scenarios is implemented in the benchmark set by the distribution of operation processing times. They do not occupy the machine capacities uniformly as in most of the standard benchmarks [30]. This is shown by the deviation σ_u of utilising the machines. An above average utilised machine will probably turn out as a bottleneck during the scheduling process. The last column of the table refers to the deviation of job-weights σ_w indicating a different distribution of job priorities in the problems. All this verifies a strong heterogeneity of the scenarios of the benchmark set.

3 Priority Rule Based Dispatching

Recent scheduling systems predominantly rely on priority rules which are found to work favourable by experience. These components of shop-floor control have been intensively studied over the last decades. Roughly spoken, each time a machine gets idle, a dispatcher selects a job among those awaiting processing. This decisions is taken according to a priority rule.

3.1 Priority Rules

Among the large number of rules proposed [25, 4], we concentrate on three rules which are known to reduce job flow-times and job tardiness effectively [20]. The FCFS rule is considered for reasons of comparison.

Table 2: Average performance of probabilistic dispatching for different objectives. The relative improvements obtained against deterministic FCFS dispatching are shown. Superior measures appear in grey shade.

rule	\overline{F}	\overline{T}	T_n	T_{\max}
FCFS	0.000	0.000	0.000	0.000
FCFS _p	0.046	0.099	0.097	0.135
S/OPN _p	0.023	0.170	0.013	0.327
SPT _p	0.083	0.198	0.205	0.075
COVERT _p	0.051	0.217	0.129	0.269

FCFS. The first-come, first-serve rule attempts to implement a fair conflict solver. The outcome of this rule is close to what can be expected from reasonable decisions while neglecting any particular properties of jobs.

SPT. The shortest processing time gives priority to that operation with the shortest imminent processing time. Jobs waiting in a queue may cause that their dedicated successor machine runs idle. SPT alleviates this risk by reducing the length of the queue in the fastest possible way.

S/OPN. The slack of a job is defined as the time span left within its allowance, assuming that the remaining operations are performed without any delay. Since jobs may wait in front of each machine the rule “slack per number of operations remaining” gives priority to the job with the minimum ratio of slack and the number of remaining operations.

COVERT. The cost-over-time rule combines ideas of SPT and S/OPN. It prioritises jobs according to the largest ratio of the expected job tardiness and the required operation processing time. In this way it retains SPT performance but seeks to respect the due dates if jobs are late, compare [28].

Simple modifications of the above rules enable the explicit consideration of job-weights [31].

3.2 Probabilistic Dispatching

A dispatcher produces a single schedule on the basis of a certain priority rule. According to [3], a more balanced assessment of a rule results from applying it in a probabilistic fashion. For this end the priorities assigned to conflicting operations are used for stochastic sampling. While doing so, multiple schedules can be produced from one rule by retaining its original character.

To reach first insight into the difficulty of the scenarios the above described rules are applied probabilistically to every instance for 1,000 times. In order to report aggregate measures for the entire benchmark set, the best result gained from the 1,000 runs for a certain problem is set in relation to the outcome of the deterministic version of FCFS rule. These values, averaged over the

twelve problems, are shown in Table 2 with respect to the four criteria under consideration.

The results shown for FCFS verify that a probabilistic dispatcher can produce considerable improvements. We further confirm that SPT is the dominating rule if the minimisation of \bar{F} is pursued. This rule also works quite well for reducing T_H . However, for the other due-date oriented criteria SPT performs rather weak because it tends to delay those jobs badly which require comparably large processing times on at least one machine. S/OPN works favourable for minimising T_{\max} . Here, SPT fails completely and is even outperformed by probabilistic FCFS. In order to minimise \bar{T} the COVERT rule is most suitable. In summary we can state that probabilistic dispatching improves the performance of a rule significantly. Nevertheless a general purpose rule is not available which motivates further research concerning adaptive scheduling techniques.

4 Adaptive Scheduling Algorithm

In this section we subsequently describe GA components, all of them already known from previous research in Operations Research and Evolutionary Computation. First, the encoding and decoding of schedules is sketched before the genetic operators are briefly described. Afterwards, the resulting algorithm is applied to the benchmark set and its performance is compared with the outcome of probabilistic dispatching.

4.1 The Genetic Algorithm

Literature reports many GA approaches to scheduling, for an overview compare [10, 11, 22, 26]. An intensive discussion has comprised the design of problem representations and search operators. Since job shop scheduling is actually a constrained multi-sequencing problem, a lot of research has concentrated on the encoding of operation sequences of machines. As a drawback, difficult crossover operators have to be used in order to enforce the feasibility while recombining the operation sequences.

Schedule Encoding and Decoding. Instead of operation sequences we encode a schedule through priorities of operations. The control of the dispatcher is driven by priorities among operations which are represented as a permutation of the operations involved in a problem. Whenever two or more operations compete for an idle machine, the one is given priority which occurs leftmost in the permutation. In this way a permutation is used like a complex priority rule.

Like with rule based dispatching, machine schedules are generated such that no machine is kept idle when it could start processing some operation. Therefore these schedules are referred to as non-delay schedules. Unfortunately, there are not necessarily optimal schedules in the set of non-delay schedules regarding a regular measure of performance. In general, a measure is defined regular if the objective function value is non-decreasing in the job completion times [14]. Suppose two schedules where the jobs complete no later in the first schedule than in the second schedule. Then, under a regular measure of performance, the first schedule is as least as good as the second.

parent permutation 1	A	B	C	D	E	F
parent permutation 2	C	A	B	F	D	E
select parent no. (1/2)	1	2	1	1	2	2
offspring permutation	A	C	B	D	F	E

Figure 1: Precedence Preservative Crossover for a problem consisting of six operations A-F.

Since \bar{F} , \bar{T} , T_{\max} and T_n are all regular measures of performance, optimal schedules can always be found in the set of active schedules. In active schedules, no operation can be started earlier without delaying some other operation or violating the machine routings. Active schedules are achieved by a slightly modified dispatching procedure which incorporates the classical algorithm of Giffler and Thompson [17].

Crossover. Since many sequencing and assignment problems can be encoded by permutations various permutation crossover operators have been developed [13]. Depending on the particular type of problem, the preservation of absolute or relative order appears advantageous. For problems which combine sequencing and assignment tasks, like vehicle routing, the preservation of precedence relations is assumed favourable [5]. This idea is adopted for scheduling problems leading to the precedence preservative crossover operator (PPX) [7].

This operator is used in our GA and is therefore described in detail, cf. Figure 1. A binary vector of equal length as the permutation is filled at random. This vector defines the order in which the operations are successively drawn from parent 1 and parent 2. We now consider the parent and offspring permutations and the binary vector as lists. We start by initialising an empty offspring. Then the leftmost operation in one of the two parents is selected in accordance to the leftmost entry in the binary vector. After an operation is selected it is deleted in both parents and appended to the offspring. Finally the leftmost entry of the binary vector is also deleted. This procedure is repeated until the parent lists are empty and the offspring list contains all operations involved.

4.2 GA Based Scheduling

Note that the PPX operator passes on precedences among operations given in two parental permutations to one offspring at the same rate, while no new precedences between operations are introduced.

Mutation. In our GA we alter 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 precedence relations of one operation to all other operations are affected, but typically a mutation has a much smaller effect.

The components described above are embedded into the framework of a simple GA [18]. The measures of schedule performance considered in this paper refer to the minimisation criteria (1)-(4). Therefore the selection mechanism of

Table 3: Comparison of performance between probabilistic dispatching and two variants of adaptive scheduling.

schedule quality	\overline{F}	\overline{T}	T_n	T_{\max}
best probabilistic rule	0.083	0.217	0.205	0.327
mean GA (non-delay)	0.095	0.279	0.259	0.314
mean GA (active)	0.089	0.294	0.295	0.322
standard deviation				
best probabilistic rule	0.02	0.07	0.06	0.11
mean GA (non-delay)	0.01	0.02	0.02	0.03
mean GA (active)	0.02	0.04	0.04	0.05
average runtime (sec.)				
GA (non-delay)	14	15	7	9
GA (active)	30	30	9	14

our GA is based on inverse proportional fitness. The PPX operator is applied with probability 0.6, whereas the mutation operator is applied with probability 0.01. A fixed population size of 100 individuals is used.

After a permutation has been decoded, the permutation is rewritten with the sequence of operations as it has been actually dispatched. In doing so a faster convergence is enforced. In order to take advantage from a fast convergence, a flexible termination criterion is used. The algorithm terminates after T generations carried out without gaining further improvements. We confine T to one half of the number of operations contained in a scenario. In this way the algorithm is given a reasonable longer time to converge for larger problems.

The GA is run in two variants, using a non-delay and an active dispatcher. Both algorithms are run for 50 times for every benchmark problem and every objective considered. The results obtained are aggregated like already described in Section 3. Table 3 shows the mean quality, deviation, and runtime of these computations (measured on a Pentium/200 Mhz). The mean quality is shown together with the results obtained by the best performing rule, which is taken from Table 2.

It can be seen that both GAs adapt towards requirements of the different objectives. With the only exception of T_{\max} , they outperform the most suitable priority rule in every case. Improvements range up to 7.5% for \overline{T} and to 9% for T_n . However, the improvement for \overline{F} is not remarkable, since already SPT performs excellent for this objective. Regarding T_{\max} neither the non-delay GA nor the active GA is able to produce a schedule comparably good as S/OPN. In summary, for two out of the four objectives a simple GA can hardly outperform a priority rule.

Comparing both GA variants, the active GA turns out superior merely for three criteria. In minimising \overline{F} the non-delay GA is clearly advantageous. This finding strikes because non-delay schedules form a true subset of the active schedules. Conversely, there may exist active schedules of better quality than the best non-delay schedules. Although within the scope of search, solutions found by the active GA for \overline{F} are worse than those found by the non-delay GA. Here adaptation fails to explore the larger space of active schedules.

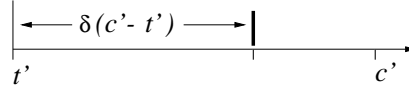


Figure 2: Dispatching interval on machine M' with $\delta = 2/3$.

The standard deviation of the GA results is pleasantly small if compared with probabilistic scheduling. A small deviation is considered advantageous because an adaptive search method, different than a priority rule, will be applied only once under realistic conditions. We have reflected this handicap by reporting the mean schedule quality achieved. The observed standard deviation consistently duplicates when switching from non-delay to active dispatching.

Due to the flexible termination criterion used, the active GA is more time consuming on average than its non-delay counterpart. This is explained by the larger search space, explored under an active dispatcher. The increase of computation time corresponds to the increase of the standard deviation observed. The computation time of the probabilistic dispatcher is of course much shorter. It solely depends on the number of iterations carried out. By increasing this number we can approximate the GA runtime, but we will hardly produce significant improvements anymore.

The robustness of adaptive scheduling concerning the different measures of performance appears highly attractive. Although quite successful if compared with priority rule based dispatching, it must be doubted whether the GA results justify the effort spent so far. Improving the GA capabilities while keeping the flexibility and broad applicability of the algorithm is the subject of the reminder of this article.

5 Scaling the Search Space

The common way to improve GA performance focuses on tuning its parameters, i.e. to enlarge the population size, to alter the selection pressure, etc. If all the parameters are already within useful bounds, the expected improvement usually does not justify the costs of finding an even more appropriate fine tuning. More universal approaches address the population management, i.e. using elitist strategies, structured populations etc. Although coming along with some improvements, such techniques are certainly not able to eliminate the problem immanent weakness of adaptation. More substantial improvements require an adjustment of the scope of search, by e.g. modifying the encoding or by intensifying the decoding procedure.

5.1 A Tunable Dispatcher

Following the latter direction it has been noticed in [12] that a non-delay dispatcher sometimes produces much better results than an active one. In Table 3 we have made a similar observation for \bar{F} . This discovery motivates the idea to vary the scope of search by means of a tunable dispatcher which scales the range between non-delay and active schedules.

Table 4: GA performance using a tunable dispatcher.

δ	\overline{F}	\overline{T}	T_n	T_{\max}
0.0	0.095	0.279	0.259	0.314
0.2	0.104	0.319	0.276	0.345
0.4	0.111	0.335	0.291	0.356
0.6	0.113	0.349	0.314	0.372
0.8	0.106	0.334	0.308	0.353
1.0	0.089	0.294	0.295	0.322

Let t' denote the earliest possible starting time of an operation on machine M' and c' the earliest possible completion time of an operation on that machine. In the Giffler and Thompson algorithm the set from which an operation is selected for dispatching (critical set) is defined by those operations which can start at time $t \in [t', c']$. Dispatching operations from the critical set always lead to an active schedule. By reducing the time span from $[t', c']$ to $[t', c'']$ with $c'' < c'$, the set of addressable schedules decreases towards the non-delay schedules.

In order to control the dispatcher, we introduce a parameter $\delta \in [0, 1]$ which defines a bound on the time span a machine is allowed to remain idle. At the extremes $\delta = 0$ produces a non-delay schedule, while $\delta = 1$ produces an active schedule. Now, the critical set of operations refers to those which can start at time $t \in [t' - \delta(c' - t'), c']$ as illustrated in Figure 2. For details of an efficient implementation of this algorithm in order $O(n^2)$ see [29]. The appropriate setting of the dispatching parameter δ within our GA remains as an open issue.

5.2 Computational Study

In order to evaluate the tunable dispatcher we use the same experimental setup as described in Section 4. The GA employs the dispatcher where δ is taken from $[0, 1]$ in steps of 0.2. The corresponding results are shown in Table 4. Notice that the first and the last line (non-delay schedules with $\delta = 0.0$ and active schedules with $\delta = 1.0$) already appear in Table 3.

The most striking result is that solutions, much better than those obtained from non-delay and active scheduling, are found in every case. Regardless of the particular objective pursued, a value of $\delta = 0.6$ yields the best mean performance for the entire benchmark set. Notice that even the maximal tardiness criterion where S/OPN has produced the best result thus far ($T_{\max} = 3.27$), is improved clearly by 4.5%. SPT, so powerful concerning the mean flow-time criterion ($\overline{F} = 0.83$) is outperformed by at least 3%.

With respect to the heterogeneity of the benchmark set, it can be conjectured that for some problems parameters different than $\delta = 0.6$ work best. To investigate this issue we take a closer look at problems no. 7, 10 and 12 while varying δ for the mean tardiness criterion. In order to achieve comparability for the three problems, the measure is normalised in the following way. The worst average performance, produced in 50 GA runs under a particular dispatching parameter, is assigned a normalised measure of 0.0. Accordingly, the

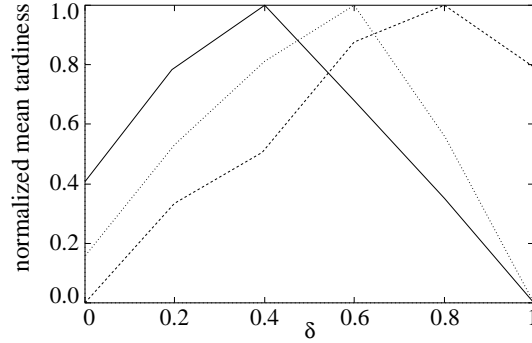


Figure 3: GA performance, depending on δ , for three selected scenarios.

best average performance is assigned a normalised measure of 1.0. The further results obtained by varying δ are linearly scaled. This leads to the plot shown in Figure 3.

The figure confirms the conjecture that the reliability of the GA reduces with an increasing search space. Starting from non-delay dispatching with $\delta = 0$, the increase of δ improves the quality for all three problems. Previously excluded solutions are now included into the scope of the search while the GA still works sufficiently well. Beyond a certain value of δ , a further increase of the search space leads to a deterioration of the solution quality.

The above finding strikes because increasing δ potentially introduces solutions of better quality into the search space. However, increasing δ incorporates a few solutions of superior quality at the expense of including a vast majority of inferior solutions into the scope of search. If δ comes close to 1 the GA neither finds newly included solutions of superior quality nor is it able to identify solutions which have already been found with smaller values of δ .

We find different values of δ to be most suitable for the three scheduling scenarios under consideration. These critical values, where the capabilities of the GA are exhausted, indicate the specific intractability of a scenario. The more difficult a problem is, the smaller the critical value of δ gets. Unfortunately, the difficulty of a problem is unknown in advance. Therefore the choice of an appropriate dispatching parameter remains as a matter of experience [23].

The computational demand of the GA also depends on the parameterisation of the dispatcher. Using $\delta = 1$ yields the largest search space which requires the longest computation time. Accordingly, $\delta = 0$ leads to a significantly shorter computation time. The average runtime needed for a particular objective under a varying δ can be taken from Table 3 by a linear interpolation of the values shown in the two bottom lines.

To summarise, we have seen that decoding active schedules within an adaptive framework does not always yield superior results. The reliability of adaptation obviously depends on the complexity of the space to be searched. Fortunately, the scope of this space can be scaled continuously. This feature enables the GA to find schedules of satisfying quality at reasonable computational costs even for complex scheduling scenarios.

6 Temporal Problem Decomposition

In practise, another way of pruning the search space is known since long. Optimisation of management tasks often involve complex models and large data sets which cannot be solved adequately. Therefore decomposition approaches are acquired which divide an overall planning problem into solvable optimisation problems. A temporal decomposition into planning periods is an appropriate way to treat large scheduling problems. Jobs are grouped according to their release-dates into daily, weekly or monthly planning periods. Then the various subproblems, each comprising a certain period, are solved consecutively on a rolling time basis. In the following we apply this decomposition technique to the scheduling scenarios to gain a further reduction of the search space.

6.1 Generating Subproblems

In order to achieve a temporal decomposition for a dynamic scheduling problem the jobs are enumerated according to their release dates. This sequence is divided into ϵ periods (degree of decomposition) such that all periods contain approximately the same number of jobs. The jobs of the first period belong to the first subproblem of the overall scheduling problem. Solving the first subproblem leads to a table of starting times for the operations involved. The second subproblem consists of all jobs which are released in the second period. Additionally, it comprises those operations of the first subproblem which have starting times larger than the release date of the first job in the second subproblem.

The detailed steps of generating a sequence of subproblems over time are summarised below. Here t denotes the earliest release date of a job in a planning period.

1. Remove all operations started before t from being considered.
2. Remove a job if all of its operations have been removed already.
3. Adjust the release-dates of jobs begun but not completed.
4. Adjust the earliest time of availability for machines busy at t .
5. Add all jobs released in the current period to the problem data.

The control structure of GAs as depicted in Figure 4 allows a straightforward integration of the above procedure [9]. The production process is viewed as being loosely coupled with the adaptive search process performed by the GA. This search is suspended after the GA has met the flexible termination criterion. The best solution of the population is used as a schedule for the period. After its implementation the GA population is adjusted according to the requirements of the subproblem of the next period. This procedure requires two simple steps.

1. Operations which are no longer part of the current subproblem are deleted in all permutations of the GA population.
2. Operations of jobs released in the next period are randomly inserted in all permutations with respect to their machine routing.

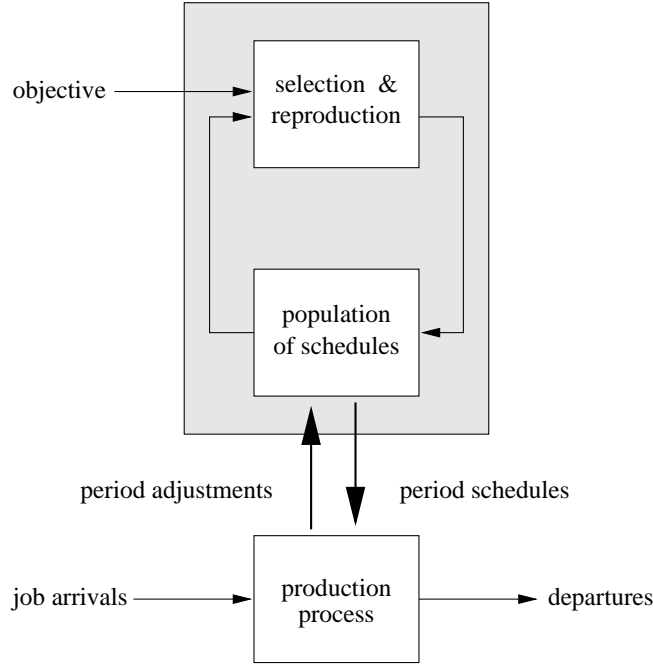


Figure 4: The GA control structure for dynamic production scheduling.

The population of schedules serves as a memory for the production process. Different to other algorithms which might be used for solving the subproblems consecutively, the GA starts searching from the scratch in the first period only. In later stages, the already scheduled backlog of a period is used to bias the adaptation process for the next period.

6.2 Computational Study

For a computational study the scenarios presented in Table 1 are stressed again. The decomposition degree ϵ , used to solve a scenario, affects the size of the space that has to be searched in the consecutive subproblems. Obviously, the larger ϵ is, the smaller the search spaces get. Using a very large decomposition degree leads to tiny subproblems which can be solved effectively. At the same time this large decomposition degree reduces the horizon of planning which can deteriorate the overall schedule quality again. In order to discover a reasonable trade-off between the size of the subproblems and the planning horizon, we vary the degree of decomposition by $\epsilon = 1, 2, 3, 6$ for the scenarios.

Like before the dispatcher is parameterised from non-delay scheduling ($\delta = 0$) to active scheduling ($\delta = 1$) by steps of 0.2. Again, the GA is run for 50 times on the twelve scenarios and the four different objectives each. As in the previous computations a value of $\delta = 0.6$ works best on average. The aggregated results obtained for this setting are shown in Table 5 with respect to a variable degree of problem decomposition.

Under $\epsilon = 1$ the test problems are solved at a whole. The corresponding

Table 5: GA performance using a variable degree of problem decomposition. The dispatcher is parameterized with $\delta = 0.6$.

ϵ	\overline{F}	\overline{T}	T_{n}	T_{max}
1	0.113	0.349	0.314	0.372
2	0.115	0.348	0.319	0.382
3	0.115	0.345	0.313	0.384
6	0.114	0.340	0.300	0.381

results in the first line of Table 5 are already known from the line shown for $\delta = 0.6$ in Table 4. It strikes that a larger ϵ hardly effects the solution quality. In some cases a higher degree of problem decomposition even increases the quality, although the increase is not significant. The loss of quality caused by decomposing a scenario and the gain reached by supporting the GA capabilities are almost balanced. The adequate tuning of the dispatcher has already pruned the search space such that a benefit due to an additional decrease of the search space is limited.

Also a loss of quality which is caused by the problem decomposition appears to be limited. The amount of backlog passed from subproblem to subproblem can be taken as a clue on the potential losses. The backlog rate is high in congested shop floors, a situation which is modelled via a high utilisation rate. For the scenarios considered, the mean utilisation rate $U = 0.83$ is rather moderate, cf. Table 1. Even under $\epsilon = 6$ the average backlog rate observed comprises about 10% of the operations belonging to a subproblem. This pleasantly low percentage explains the results obtained.

Let us now have a look at the computation times needed to solve the scenarios under a variable degree of problem decomposition. We have already seen that computation times increase approximately linear with an increase of the parameter δ . Recall that increasing δ leads to an enlargement of the search space. Increasing ϵ will have an opposite effect as it actually reduces the scope of search with a higher degree of problem decomposition. Thus one can expect savings of runtime by increasing the parameter ϵ .

The above conjecture is confirmed by Figure 5 which shows the average GA runtime for the scenarios with respect to the different combinations of δ and ϵ . The figure presents the measures taken for \overline{T} . The relation between the individual measures are transferable also to the three other objectives considered. We already know from Table 3, that \overline{F} requires more or less identical computation times. The measures for the remaining objectives T_{n} and T_{max} approximately halve the height of the columns.

The computation takes about thirty seconds for ϵ and δ both set to one. A decrease of δ as well as an increase of ϵ shorten the computation time to a fraction of the original amount. Cutting a problem into two subproblems already effects a gain larger than the one obtained by changing from an active dispatcher to a non-delay one. The issue of computation times by far justifies the temporal problem decomposition.

Summarising, we end up with an adaptive algorithmic setting, which is much more efficient compared to the original GA approaches. Against a simple GA

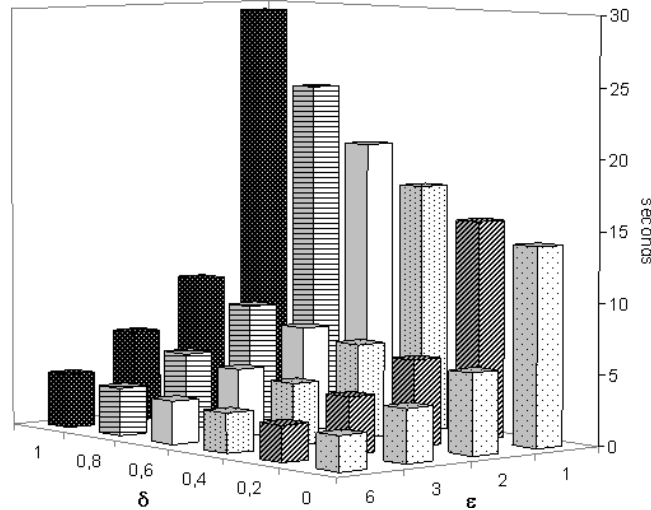


Figure 5: Average GA computation time (sec.) needed under different parameterisations to solve a single scenario. Results are shown exemplary for the \bar{T} objective.

a strong improvement of quality is observed in all cases while cutting down the computation time considerably. Although setting the parameters appropriately remains as a matter of experience, we have shown that the algorithm is robust concerning the parameterisations of δ and ϵ .

7 Relation to Previous Approaches

In order to assess the improvements gained so far we compare the performance of our GA with other evolutionary approaches to dynamic job shop scheduling. In two recent papers [16, 21] six of the twelve scenarios have been tackled. Therefore we are able to compare the best results found over the entire computational study with results published in literature.

7.1 Discussion of Approaches

Beside priority rule based dispatching, Fang et al. have applied two different adaptive approaches to the scenarios. The first method is a stochastic hill-climbing algorithm which works quite similar to an Evolutionary Strategy with one parent and one offspring. This algorithm performs 35,000 schedule evaluations. As a second method, a GA previously described in [15] is used. To provide a universally applicable algorithm, tailored operators are omitted in this approach in favour of standard techniques such as a permutation representation of machine sequences and the uniform crossover operator. The GA also uses a flexible termination criterion, nevertheless the GA effort is limited to 100,000 evaluation performed.

In contrast to Fang et al., Lin et al. have designed a highly sophisticated GA. This algorithm takes advantage from a whole bunch of evolutionary computation

techniques, such as a direct problem representation which encodes the operation starting times, tailored genetic operators, and a structured population model. To cope with the high computational demand of 1.125 million evaluations, the algorithm is parallelised.

7.2 Comparison of Results

Since both previous studies present results merely for six out of the twelve scenarios, we confine ourselves in the same way (the scenarios not listed here process two or more operations of the same job on a certain machine). For the reminder it is important to recall that scenario 1 and 2 consist of less than one third of the operations to be scheduled in the other four scenarios 7, 9, 10 and 12, see Table 1. We therefore distinguish between “small” and “large” problems.

According to the computation time reported in Table 3 we can divide the objectives into two groups, namely the mean-objectives $\{\bar{F}, \bar{T}\}$ and $\{T_n, T_{\max}\}$. As mentioned before, the computation time of the former group is about twice as long as the one observed for the latter group. In other words, the GA generates about twice the number of improvements for the former group. This indicates the particular difficulty of pursuing the $\{\bar{F}, \bar{T}\}$ objectives.

Table 6 reports the best solutions found by Fang, Corne and Ross (FCR), Lin, Goodman and Punch (LGP), and Mattfeld and Bierwirth (MB). We also list the best outcome of probabilistic priority rule dispatching (RULE).

Let us first have a look at the results achieved by FCR. Regarding $\{T_n, T_{\max}\}$ FCR outperforms the best performing rule for every scenario. When pursuing $\{\bar{F}, \bar{T}\}$ only the small problems can be improved. For all large problems FCR is inferior to the best performing rule, which again indicates that pursuing a mean-criterion in job shop scheduling is a much more difficult optimisation task compared to one of $\{T_n, T_{\max}\}$. For the small problems the best schedules of mean-criteria have been obtained by the stochastic hill-climbing algorithm.

Summarising, there is a strong indication that FCR is not able to cope with difficult dynamic scheduling problems. Although we disagree concerning certain GA components used by Fang et al., we must admit that their approach is quite similar to our simple GA. Recall from Table 3, that also our GA has not outperformed the best performing rule in every case. Thus, we state that simple adaptation is a weak schedule optimisation technique which requires a substantial support.

Lin et al. report that the small scenarios are solved by LGP at almost the same level of solution quality which has been already achieved by FCR. Thus we suppose that the small scenarios are solved to near-optimality by all GA approaches under consideration. The large scenarios are improved significantly by LGP in every case. Hence, using advanced evolutionary techniques appears worth the higher computation times spent.

The GA approach proposed in this paper does not improve any of the small problems. Probably large populations and sophisticated operators are needed in order to gain further improvements whenever a near-optimal level of quality has been reached. Turning to the large problems it is verified that a substantial progress is still possible. For both objectives of the $\{T_n, T_{\max}\}$ group three out of four scenarios are improved by MB.

scheduling scenarios are improved significantly. We conclude that a scaling of the search space can be much more beneficial for GA search than a sophisticated

Table 6: Performance of different approaches for a subset of the benchmarks presented in Table 1. The best found schedules, given in terms of the improvement-rate against deterministic FCFS dispatching, are listed for the various objectives.

critereon	problem no.	RULE	FCR	LGP	MB
\bar{F}	1	0.029	0.099	0.099	0.099
	2	0.156	0.182	0.182	0.182
	7	0.088	0.075	0.120	0.131
	9	0.179	0.166	0.229	0.254
	10	0.095	0.084	0.107	0.140
	12	0.117	0.109	0.145	0.182
\bar{T}	1	0.089	0.380	0.388	0.379
	2	0.442	0.475	0.491	0.484
	7	0.451	0.404	0.670	0.688
	9	0.334	0.272	0.486	0.499
	10	0.308	0.352	0.374	0.428
	12	0.393	0.361	0.473	0.568
T_h	1	0.167	0.334	0.428	0.423
	2	0.451	0.550	0.551	0.545
	7	0.334	0.429	0.627	0.618
	9	0.413	0.591	0.688	0.750
	10	0.154	0.300	0.319	0.315
	12	0.358	0.609	0.615	0.638
T_{\max}	1	0.383	0.596	0.585	0.596
	2	0.591	0.666	0.666	0.665
	7	0.478	0.500	0.646	0.736
	9	0.383	0.455	0.574	0.543
	10	0.442	0.455	0.526	0.539
	12	0.660	0.656	0.685	0.713

outline of the algorithm. Furthermore, take into consideration that the number of evaluations performed by LGP are on average thirty times the costs of MB. Considering this enormous gap, it strikes that our GA can produce superior schedules at all. We take this as a strong indication that even simple adaptation can succeed if applied adequately.

Definite statements about the efficacy of the compared approaches cannot be drawn. The three studies are devoted to completely different research goals. Moreover, the number of the scenarios considered is very small and the number of runs performed are not comparable, neither mean results nor deviations are presented in the cited papers. However, the figures presented in this paper provide a basis of comparison for further research on the benchmarks and may shed some light on the potentials of the approach presented.

8 Conclusions

In this paper we have dealt with some aspects of realistic scheduling scenarios and with related performance criteria. We assume that similar scheduling scenarios occur in shop floors. The heterogeneity of data, constraints and objectives have so far at least hindered the implementation of recent algorithmic approaches. GAs offer the opportunity to combine them with traditional dispatching procedures resulting in efficient optimisation techniques.

We have shown that the capabilities of GAs to conduct a proper search vanish with an increasing problem size. This inherent weakness can be alleviated by a tunable dispatching procedure which proportions the size of a search space within useful bounds. Although this technique may exclude near optimal and even optimal solutions from being considered, the positive effects of reducing the search space do prevail.

A reliable GA-based scheduling system may accept the exclusion of optima in order to keep the GA working properly on an adequately sized search space. Whenever the search space exceeds a certain size, the level of solution quality achieved does not only stagnate but tend to decrease drastically with a growing scope of search. It remains questionable if computation power and algorithmic sophistication can make up for the weakness of adaptation. On the other hand, if adequately tuned even simple genetic search can proceed.

In practice scheduling of activities are performed periodically on a rolling time basis. This temporal decomposition of a scheduling problem can be exploited by a GA to decrease the computation time without significant loss of solution quality. If a production environment changes frequently, unforeseen events can prevent to execute a schedule any further. From this point of view the construction of a long term schedule hardly makes sense. Instead treating a problem in shorter terms by problem decomposition leads to reactive scheduling system.

References

- [1] Adams, J., Balas, E., Zawack, D., 'The Shifting Bottleneck Procedure for Job Shop Scheduling', *Manage. Sci.*, **34**, 391-401 (1988).
- [2] Anderson, E. J., Glass, C. A., and Potts, C. N. 'Machine scheduling', In Aarts, E. H. L. and Lenstra, J. K. (eds.), *Local Search Algorithms in Combinatorial Optimization*, Wiley, New York, 361-414, 1997.
- [3] Baker, K.R., *Introduction to Sequencing and Scheduling*, Wiley, New York, 1974.
- [4] Blackstone, J.H., Phillips D.T., Hogg G.L., 'A State-of-the-Art Survey of Dispatching Rules for Manufacturing Job Shop Operations', *Int. J. Prod. Res.*, **20**, 27-45 (1982).
- [5] Blanton, J. L. and Wainwright, R. L., 'Multiple vehicle routing with time and capacity constraints using genetic algorithms', In Forrest, S. (ed.), *Proc. 5th Int. Conf. on Genetic Algorithms*, Morgan Kaufmann, San Mateo, 452-459, 1993.

- [6] Błażewicz, J., Domschke, W., Pesch, E., 'The job shop scheduling problem: Conventional and new solution techniques', *Eur. J. Oper. Res.*, **93**, 1-30 (1996).
- [7] Bierwirth, C., Mattfeld, D., Kopfer, H., 'On permutation representations for scheduling problems', In Voigt, H.-M., et al. (eds.), *Proc. of Parallel Problem Solving from Nature IV*, Springer, Berlin, 310-318, 1996.
- [8] Bierwirth, C., Mattfeld D.C., 'Production Scheduling and Rescheduling with Genetic Algorithms', *Evol. Comp.*, **7**, 1-17 (1999).
- [9] Bierwirth, C., *Adaptive Search and the Management of Logistics Systems*, Kluwer, Boston, to appear.
- [10] Cheng, R., Gen, M., Tsujimura, Y., 'A Tutorial Survey of Job-Shop Scheduling Problems Using Genetic Algorithms', *Comp. Ind. Engng.*, **30**, 983-997 (1996).
- [11] Corne, D., *Applied Evolutionary Computation - Solving Industrial and Scientific Optimisation Problems*, Springer, London, to appear.
- [12] Della Croce, F. D., Tadei, R., Volta, G., 'A Genetic Algorithm for the Job Shop Problem', *Comp. Oper. Res.*, **22**, 15-24 (1995).
- [13] Fox, R.B., McMahon, M.B., 'Genetic Operators for Sequencing Problems', In Rawlins, G.J.E. (ed.), *Foundations of Genetic Algorithms*, Morgan Kaufmann, San Mateo, 284-300, 1991.
- [14] French, S., *Sequencing and Scheduling*, Ellis Horwood, Chichester, 1982.
- [15] Fang, H.-L., Ross, P., Corne, D., 'A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems', In Forrest, S. (ed.), *Proc. 5th Int. Conf. on Genetic Algorithms*, Morgan Kaufmann, San Mateo, 375-382, 1993.
- [16] Fang, H.-L., Corne, D., Ross, P., 'A genetic algorithm for job-shop problems with various schedule quality criteria', In Fogarty, T. (ed.), *Proc. of AISB Workshop*, Springer, Berlin, 39-49, 1996.
- [17] Giffler, B., Thompson, G., 'Algorithms for solving production scheduling problems', *Oper. Res.*, **8**, 487-503 (1960).
- [18] Goldberg, D.E., *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison Wesley, Reading, 1989.
- [19] Hart, E., Ross, P., Nelson, J., 'Solving a Real-World Problem Using an Evolving Heuristically Driven Schedule Builder', *Evol. Comp.*, **6**, 61-80 (1998).
- [20] Haupt, R., 'A Survey of Priority Rule-Based Scheduling', *OR Spektrum*, **11**, 3-16 (1989).
- [21] Lin, S.-C., Goodman, E.D., Punch, W.F., 'A Genetic Algorithm Approach to Dynamic Job Shop Scheduling Problems', In Bäck, T. (ed.), *Proc. 7th Int. Conf. on Genetic Algorithms*, Morgan Kaufmann, 481-489, 1997.

- [22] Mattfeld, D.C., *Evolutionary Search and the Job Shop*, Physica/Springer, Heidelberg, 1996.
- [23] Mattfeld, D.C., 'Scalable Search Spaces for Scheduling Problems', In Banzaf, W. et al. (eds.), *Proc. of the Genetic and Evol. Comp. Conf.*, Morgan Kaufmann, 1616-1621, 1999.
- [24] Morton, T.E., Pentico, D.W., *Heuristic Scheduling Systems*, Wiley, Chichester, 1993.
- [25] Panwalkar, S.S., Iskander W., 'A Survey of Scheduling Rules'. *Oper. Res.*, **25**, 45-61 (1977).
- [26] Portman, M.C., 'Genetic Algorithms and Scheduling: A State of the Art and some Propositions', *Workshop Proc. on Prod. Planning and Control*, Mons, Belgium, 1-14, 1996.
- [27] Raman, N., Talbot, F.B., 'The job shop tardiness problem: A decomposition approach', *Eur. J. Oper. Res.*, **69**, 187-199 (1993).
- [28] Russell, R.S., Dar-El, E.M., Taylor B.W., 'A Comparative Analysis of the COVERT Job Sequencing Rule using various Shop Performance Measures', *Int. J. Prod. Res.*, **25**, 1523-1540 (1987).
- [29] Storer, R., Wu, D., Vaccari, R., 'New Search Spaces for Sequencing Problems with Application to Job Shop Scheduling', *Manage. Sci.*, **38**, 1495-1509 (1992).
- [30] Taillard, E.D., 'Benchmarks for Basic Scheduling Problems', *Eur. J. Oper. Res.*, **64**, 278-285 (1993).
- [31] Vepsalainen, A.P.J., Morton, T.E., 'Priority Rules for Job Shop with Weighted Tardiness Costs', *Manage. Sci.*, **33**, 1035-1047 (1987).