

A Case Study of Operational Just-in-Time Scheduling using Genetic Algorithms

Ivo Rixen, Christian Bierwirth, Herbert Kopfer

Department of Economics, University of Bremen

Abstract. Genetic Algorithms (GAs) have shown to fit the complex needs arising from many tasks of academic optimization for almost 20 years. This paper investigates the application of a GA to a real-world scheduling problem. The objective under consideration is just-in-time completion of jobs. First, the strategic concept of just-in-time production is operationalized according to the applicational demands. Using a real-world extract of dynamic data the proposed GA then serves as a schedule builder. A simulation study shows that the GA clearly outperforms the former way of production scheduling in this application area.

Keywords. Genetic Algorithm, just-in-time, dynamic machine scheduling

1 Introduction

Modern concepts for industrial production like *Lean* and *Just-in-Time* have shown a world-wide impact in the last decade, compare e.g. Womack et al. [1990]. One basic idea of the general philosophy emphasizes on-time delivery of sub-assemblies and final products. But the transfer of this claim to a practical implementation often remains an open challenge for present research. In many sectors of industrial production a variety of technical as well as organizational difficulties prevent an easy realization of these new concepts.

On this background the paper transforms the strategic objective of just-in-time production towards an operational goal for machine scheduling in an existing firm. Here, the objective pronounced by just-in-time leads to a function of minimizing the mean absolute lateness, i.e. the sum of earliness and tardiness of released jobs. A brief review on literature dealing with absolute job lateness is given by Fry et al. [1990]. Fry et al. propose a heuristic based on adjacent job exchange in combination with a linear programming approach in order to solve one-machine scheduling problems. Following this line of research we present an automated approach for generating machine schedules with respect to a measure of just-in-time completion of jobs. A GA capable to handle general job shop problems is involved as the basic component of the automated scheduling system. The GA solves a series of static problems which are created by the entry of certain events. Since e.g. job release times are unknown in advance, job scheduling appears as a dynamic problem. For this reason the GA has to act on a rolling time basis as proposed by Raman et al. [1989]. A comparison of the proposed automated scheduling approach with traditional scheduling is outlined by an extract of jobs actually processed in the workcenter in 1994.

2 The Application Case Study

2.1 Description of a Scheduling Environment

Throughout this paper we consider a firm that acts in the fastening sector. It produces specialized fastening tools for the motor-vehicle industry as well as different kinds of fasteners such as nails, staples and glue.

In the following we study a collator machine which connects certain types of nails to strips of collated nails fitting professional nailing tools. Jobs are given to the collator by specifying a lot-size and a certain nail type. The processing of different jobs may require changeover times which are regarded as a setup for the collator. In order to serve the market the sell department of the firm passes the demand to the manufacturing workcenter. On-time delivery of products, i.e. nailstrips, appears to be the most desirable goal in the workcenter. Early or tardy deliveries are highly discouraged, because they may lead to penalty costs. For this reason the sell department declares a due date for each job which ensures a security time span between job completion and delivery date. The workcenter schedules jobs with respect to the given time constraints only, i.e. without taking aspects of costs into account. As soon as a demand reaches the workcenter the needed raw material is ordered. The material arrives at the workcenter about a fortnight after ordering. This point in time is used as the ready time of the corresponding job.

Job id	Nail type	Income date	Ready time	Setup	Lot-size	Duration	Due date
N320	MC62	20.10.93	13.01.94	3	1200	0.0592	26.01.94
N321a	QC70	20.10.93	13.01.94	3	325	0.1606	09.02.94
N321b	QC70	20.10.93	02.02.94	3	325	0.1606	09.02.94
N322a	MC64	20.10.93	02.02.94	3	1107	0.0638	23.02.94
N322b	MC64	20.10.93	16.02.94	3	93	0.0638	23.02.94
N323a	MC65	20.10.93	18.02.94	3	1672	0.0688	23.03.94
N323b	MC65	20.10.93	11.03.94	3	528	0.0688	23.03.94
N324	MC61	20.10.93	11.03.94	3	800	0.0839	06.04.94
N333	QC70	10.12.93	06.12.93	3	34	0.1606	13.12.93
N334	QC70	10.12.93	02.02.94	3	40	0.1606	13.12.93

Table 1. Jobs for *Collator 3* (part I).

The table above shows all queuing jobs related to *Collator 3* which were not started within the year 1993. As shown by the table different nail types are processed by this machine. The setup of the collator takes about three hours if processing changes over to another nail type. Sometimes, jobs of the same nail type and incoming date appear in Table 1, e.g. N322a, N322b. This is caused by partly split arrivals of raw material which enforce different ready times of the jobs. Obviously, if a job succeeds another job of the same nail type no time to setup is necessary. The product of the lot-size and the duration defines the processing time of a job in units of industrial working hours.

From time to time, special jobs are released by the research department of the firm. These prototype jobs are processed for innovative purpose only. Although the research department declares a due date for these jobs, they have no definite date of delivery. Table 2 shows the prototype jobs N397 and N405, both characterized by a long setup and a small lot-size.

Let us now consider how scheduling of jobs is actually performed by the workcenter. Fastener products are usually produced by a single dedicated machine. Thus the workcenter is confronted with a number of one-machine scheduling problems. Let us continue to focus on *Collator 3*. For this machine a new schedule, containing all released and not started jobs, is built up from 6 to 7 times a year. The building of a new schedule is triggered by the desire to estimate the work load of machines within the near future. Next to that it can be forced by important events, e.g. a machine breakdown. Scheduling itself is made by hand. Hence it is less inspired by ideas of optimization than by expert knowledge. For example, if a job risks delay its completion may be accelerated by using double shifts. Nevertheless, wage-expensive double shifts should be introduced in urgent situations only.

2.2 An Automated Scheduling Approach

Let us summarize the dynamic scheduling environment of the manufacturing workcenter. Jobs are processed on a single machine without preemption. They are released at undetermined points in time which actually occur rarely and there is a changeover time between jobs of different types. The primary measure of schedule performance responds to just-in-time completion of jobs. Secondary, double shifts should be avoided whenever possible.

In the following we propose an automated scheduling approach which introduces two features of optimization into the daily coarse of the workcenter. This is a method which automatically generates machine schedules based on just-in-time performance and a rule which determines when to use the method. Presently the point in time when schedule building is done in the workcenter is not event driven. It is performed simultaneously for the whole workcenter about every two months because manual scheduling of a large number of machines is very time consuming. To the contrary, a fast computer based method permits the scheduling authority to follow the coarse of events in the workcenter (e.g. new job releases, machine breakdowns).

Concerning *Collator 3* Table 2 shows the chronology of such events in the workcenter. Whenever a new event entries, time has come to generate a new schedule, i.e. to determine starting times of new jobs and to update starting times of waiting jobs. Notice that for those jobs being ready but not yet started a new ready time is defined by the expected completion time of the current job processing. Doing it this way, the dynamic scheduling problem is treated as a series of static problems which are solved on a rolling-horizon basis. Obviously, the choice of a method solving this series of problems has an important impact on the schedule quality that is actually implemented in the workcenter.

Job id	Nail type	Income date	Ready time	Setup	Lot-size	Duration	Due date
<i>Event I at 3.1.94: Job transfer from 1993</i>							
... see Table 1							
<i>Event II at 8.2.94: New job releases</i>							
N377a	MC65	08.02.94	08.04.94	3	1496	0.0688	01.06.94
N377b	MC65	08.02.94	14.04.94	3	176	0.0688	01.06.94
N377c	MC65	08.02.94	11.05.94	3	528	0.0688	01.06.94
N378	MC62	08.02.94	11.05.94	3	1130	0.0592	15.06.94
N384	MC64	08.02.94	11.05.94	3	800	0.0638	29.06.94
<i>Event III at 11.5.94: New job releases</i>							
N391	QC70	11.05.94	14.06.94	3	750	0.1606	20.07.94
N392	QC65	11.05.94	18.05.94	3	500	0.0943	29.07.94
N393	MC65	11.05.94	18.05.94	3	2200	0.0688	15.08.94
N394	MC64	11.05.94	18.05.94	3	1570	0.0638	16.09.94
N395	QC70	11.05.94	18.05.94	3	330	0.1606	30.09.94
N397	RE58	11.05.94	26.05.94	14.5	10	0.1898	24.06.94
<i>Event IV at 11.7.94: New job releases</i>							
N405	RE58	11.07.94	20.07.94	14.5	50	0.1898	27.07.94
N406	MC61	11.07.94	22.07.94	3	680	0.0839	28.09.94
N412	MC65	11.07.94	22.07.94	3	2570	0.0688	23.11.94
N413	QC70	11.07.94	22.07.94	3	620	0.1606	19.10.94
N416	MC62	11.07.94	22.07.94	3	1030	0.0592	30.11.94
N419	MC64	11.07.94	22.07.94	3	740	0.0638	21.12.94
<i>Event V at 16.8.94: Machine breakdown until 5.9.94</i>							
<i>Event VI at 28.9.94: New job release</i>							
N420	QC65	28.09.94	12.10.94	3	500	0.0943	21.12.94

Table 2. Jobs for *Collator 3* (continued).

On this background the use of a GA offers an interesting opportunity outlined by Bierwirth and Kopfer [1994]. At the end of a GA run the generated schedules may contain jobs which will not be started until the next event occurs. The corresponding partial solutions can be used by the initial GA population in order to solve the problem resulting after the new event. In a situation where events follow close after each other or even real-time scheduling is required, this strategy can enormously enhance the efficiency of genetic search.

Runtime performance is of subordinate interest in favor of effectivity because events occur rarely in the presented application. Thus it seems more suitable to use random initialized populations, run the GA several times on the same static problem, and finally implement the best of all generated schedules in the work-center. Nevertheless, using a GA for schedule generation offers another important advantage. For related or even more complex scheduling problems a suitable GA is available, see Rixen and Kopfer [1994]. We can adapt it to the needs of our application problem by only modifying the schedule building procedure. The adaptation of schedule evaluation according to just-in-time completion of jobs is a topic of the next section.

3 Incorporating Genetic Algorithms

3.1 Just-in-Time Evaluation of Schedules

In order to model the overall goal of the workcenter, i.e. just-in-time completion of jobs with respect to a minimal number of double shifts, the following standard notations are used:

- J_j Variables for job id's ($j = 1, 2 \dots n$),
- r_j Ready time of J_j ,
- s_{ij} Setup time, changing over from J_i to J_j ,
- p_j Processing time of J_j in working hours (Lot-size \times duration),
- d_j Due date of J_j ,
- α_j Weight factor between earliness and tardiness of J_j ($0 < \alpha_j < 1$),
- h Industrial working hours per shift ($h = 7.5$).

In a first step scheduling of tasks on the collator machine requires to sequence n jobs. Assume such a sequence to be given, e.g. $S = J_1, J_2 \dots J_n$. Within our approach job sequences like S are generated by a GA. For evaluation the GA needs access to a procedure building a feasible schedule from a sequence and returning its measure of performance.

The question arising is how to calculate starting times for sequencing jobs on the collator. Usually machine idle times are inserted between the processing of adjacent jobs if this enhances the just-in-time measure, compare Fry et al. [1990]. For the problem considered this strategy may lead to an increasing number of double shifts if several jobs have a similar due date.

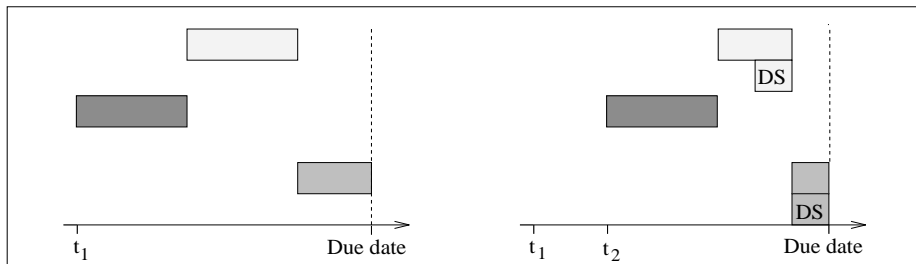


Fig. 1. Schedule chart of 3 jobs with identical due date.

Figure 1 shows the alternative between starting a job at t_1 or inserting an idle time of $t_2 - t_1$. Although the job is less early in the right chart it enforces double shifts for the following jobs not to become tardy. However, double shifts are discouraged because they increase the costs of a working hour by about 20%. They appear to be reasonable only if a job risks to be tardy. Thus we are confronted with a situation where minimizing double shifts and earliness of jobs state conflicting subgoals in combination with the dominating goal to minimize

job tardiness. Since costs arising from earliness of jobs cannot be quantified by the workcenter we minimize the number of double shifts in a prior manner. Therefore we refrain from introducing idle times to the collator, i.e. we consider left-shifted schedules of jobs. Doing it this way, the starting times t_j of jobs given in a sequence S are iteratively calculated by

$$t_j = \max \left((t_{pred(j)} + p_{pred(j)} - h \cdot N_{pred(j)} + s_{pred(j),j}), r_j \right).$$

Here, N_i denotes the number of double shifts used to process job J_i where $i = pred(j)$ refers to the predecessor of J_j in S , if it exists. Otherwise J_j is the first job of S and t_j is given by the maximum of the machine setup and the ready time only. Both, the sequence of jobs and the number of double shifts for each job are encoded in the genetic representation scheme of schedules. Notice that the term hN_i gives the sum of working hours which are planned to be done in double shifts in order to accelerate job J_i . Since J_i precedes J_j the product hN_i has to be subtracted in order to determine the actual starting time of J_j .

Given the starting time of J_j its completion time simply results from $C_j = t_j + p_j - hN_j$. Thus earliness and tardiness of jobs is calculated by $E_j = \max(d_j - C_j, 0)$ and $T_j = \max(C_j - d_j, 0)$. For some jobs it may be very important not to be late, some other jobs should not be completed too early. Therefore, a job specific weight factor α_j may be useful to express a preference between early or tardy completion. Hence we measure the just-in-time performance of a job by $JIT_j = \alpha_j E_j + (1 - \alpha_j) T_j$. If no preferences are given to a job the weight factor is set to 0.5. Notice that this setting implicitly tends to prefer a shorter tardiness at the expense of a shorter earliness caused by the building of left-shifted schedules. Nevertheless, since $0 < \alpha_j < 1$ holds, we can state J_j to be completed just in time if and only if $JIT_j = 0$. The just-in-time measure JIT_j of a prototype job J_j is often rather unimportant. In order to neglect the influence of these jobs on the mean quality of solutions their measure may be set to zero beforehand. Now, the overall goal of the workcenter can be stated as

$$\text{minimize } \overline{JIT} = \frac{1}{n} \sum_{j=1}^n JIT_j.$$

This objective function operationalizes the concept of just-in-time production with respect to the details of the application considered. Of course, the details are partly hidden in the genetic representation of the application itself instead of the fitness function. But this apparent drawback is known to be a common feature of successful genetic search in scheduling, compare e.g. the various GA approaches proposed by Pesch [1994].

3.2 Genetic Schedule Representation and Operators

We are looking for a genetic schedule representation which depicts the presented one-machine scheduling problem in the workcenter right down to its important decision details. As outlined in the previous section schedule building requires to

make at least two kinds of decisions. First of all a schedule is constructed with respect to a sequence representing the order of processing jobs on a machine. Second, a schedule is constructed by means of some integer values representing the number of double shifts for each job. Other variables used in the previous section such as completion times, earliness, and tardiness of jobs can be expressed in terms of these both decisions. For this reason we propose a genetic schedule representation of two interacting chromosomes.

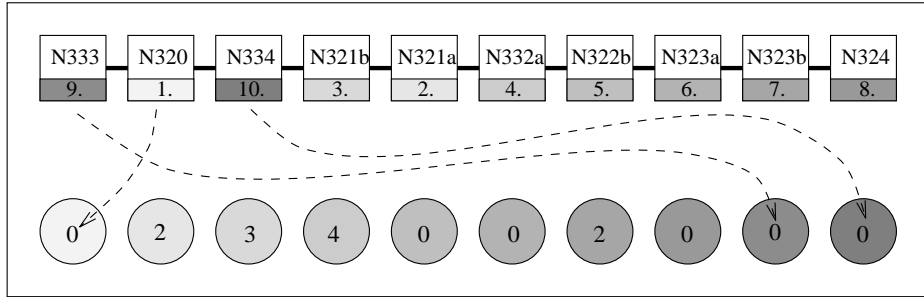


Fig. 2. Schedule decoding of an arbitrary genotype.

Chromosome 1 encodes the processing sequence of jobs by use of a permutation scheme of length n . Thus, each item of the permutation represents one job identifier. Reading a permutation from left to right hand side directly decodes a processing sequence. By this a natural representation is given which covers each feasible job sequence exactly once. An important advantage of this representation is that a wide range of genetic operators is available for permutation schemes. Nevertheless, the choice of any of these operators should take the properties of the application problem into account. Most of all permutation operators have been designed to serve as a crossover in traveling salesman applications. In this problem class the relative order of cities represented by a permutation is of dominating importance. A certain city should be visited within the surrounding of other cities. Whether this happens for example somewhere in the beginning of a tour or not is of subordinate interest. Quite to the opposite, scheduling requires a somehow position based ordering of jobs. Consider a job which has to be processed early because of an early due date. Then it should occur – independently from the relative order of other jobs – somewhere in the beginning of a permutation. Thus we use *Position-Based Crossover* as proposed by Syswerda [1991] which strongly supports the inheritance of position characteristics. Mutation of Chromosome 1 is performed by swapping of two arbitrary jobs.

Chromosome 2 encodes the number of double shifts used to process each of the n jobs. Consider again the sequence of job identifiers represented in Chromosome 1. Each job identifier J_j can be used as an index that clearly points to a field of Chromosome 2 carrying its N_j value, i.e. the number of double shifts. Hence Chromosome 2 is handled as an array of size n . Figure 2 illustrates

the procedure of decoding a bichromosomal genotype. The upper chromosome shows an arbitrary sequence of the 10 jobs sketched in Table 1. The grey scale of items refers to indexes of jobs. As indicated by the three dashed arcs each index references a field of the lower chromosome.

The integer values of Chromosome 2 range from 0 to a maximal number of double shifts of the corresponding job. The number of double shifts are respectively bounded by the integer part of $\frac{p_j}{2}$ for job J_j . The use of integers is motivated by the idea that a double shift is introduced only if its capacity is fully exploited. If $N_j = 0$ the job J_j will be processed without double shifts. Consider job N324 mentioned in Table 1. It occupies *Collator 3* with a total of $800 \times 0.0839 = 67.17$ industrial working hours. Notice that a single shift counts 7.5 industrial working hours. Hence N324 can be processed by inserting at most 4 double shifts leading to an entire processing of approximately 5 working days. Without any double shifts N324 occupies the collator machine for about 9 days.

Initial array instantiations of Chromosome 2 are randomly generated using an integer valued uniform distribution over the interval $[0, \frac{p_j}{2}]$ for index j . Since Chromosome 2 responds to the array type Two-Point Crossover performs a valid genetic operation. Furthermore, mutations are achieved by randomly selecting a field of the array and increasing or decreasing its value by 1 with respect to the feasible bounds.

To summarize recombination, two offspring are generated from the genotypes of two parents by applying *Position-Based Crossover* and *Two-Point Crossover* to Chromosome 1 and 2 respectively. Mutation of genotypes is achieved in an appropriate manner. The probability of crossover is set to 0.5 and the mutation rate to 0.001. Both parameters appear to follow an often used standard setting.

4 Simulation Study of Automated Scheduling

Following the data of Table 2 we perform an event-driven and GA-based job scheduling for *Collator 3*. The entire simulation run is divided by the six events I–VI. Each event meets a given state of the collator machine which is transformed into a static scheduling problem. Consider e.g. *Event I*. Any decision of how to schedule the 10 jobs N320–N334 determines one part of the scheduling problem we are facing at *Event II*. The other part of this new problem results from the currently released five jobs N377a–N384. Thus we are faced with a total of six scheduling problems, but only the first of them is determined in the beginning.

Throughout our simulation the GA runs on each of the six static problems for a total of 30 iterations. Using the evaluation and recombination techniques described in the previous section, the population size of the GA is set to 128 individuals and termination occurs after 100 generations. For evaluation a constant weight factor $\alpha = 0.5$ is used. Further implementation details of the GA can be found in Rixen and Kopfer [1994]. Although runtime performance is of subordinate interest we would like to mention that a single GA run requires an average runtime of 17 seconds measured on a Sun10 workstation. For each run

the best generated schedule is stored. Finally, after 30 iterations have been done the best of all schedules is implemented on *Collator 3*.

The simulation results are shown in Table 3. The jobs listed in the first column appear in the suggested order of processing. Each event is reported by a subtable giving the best GA-generated schedule of the corresponding static problem. The three columns of a subtable refer to the obtained completion time of jobs, the machine setups, and the numbers of used double shifts. A symbol "x" denotes that a machine setup is required. Notice that values representing a date are given in accumulated units of industrial working hours. The entry date *3.1.94* of *Event I* refers to the virtual starting time 0 of the simulation. In order to avoid negative time values the due date of jobs transferred from 1993 is calculated by the maximum of 0 and the original due date.

It can be seen from Table 3 that most of the jobs are involved within several static problems arising from different events. This is the case for those jobs that have not been started before the next event. Consider e.g. the jobs N323b and N323a. Within the period after *Event I* they are scheduled for completion at 427 and 405 respectively, i.e. N323a precedes N323b. But in the following period the processing order changes as shown by the new completion times of 404 and 496. Within this period both jobs are actually processed since they do not appear in the back part of the table again. Of important interest is the behavior of automated scheduling in case of such unpleasant events as machine breakdowns. Have a look at *Event V* at *16.8.94*. The date corresponds to a virtual time of 1191, i.e. the breakdown occurred while setting up the collator for Job N394. Since the machine was repaired until *5.9.94* it could not be used for a total of 32 working hours. The seemingly larger difference between both dates results from a two week vacation in the firm which is not expressed by the virtual simulation time. Notice that the processing order of the following 7 jobs does not change after the breakdown whereas the number of double shifts increases drastically.

Let us now have a look how the objectives of scheduling are satisfied in the simulation run. The bottom row of Table 3 accumulates the number of double shifts which are actually used within one period. Consider Job N377c which is first scheduled at *Event II* using a single double shift. Since its processing is not started in that period we count a total of 7 instead of 8 double shifts. Furthermore we notice that starting this job in the next period does not require a double shift at all. The right column of the table shows the just-in-time performance of each job in terms of earliness and tardiness. The only job which exactly completes at the due date is N378. For most of the other jobs an acceptable deviation of a few hours is reached. Nevertheless, at least some jobs appear to be critical with respect to the due date. Consider e.g. N334. Its completion has a delay of 171 hours, since a due date of 0 hits the beginning of the simulation run. So, why should job N320 precede N334 as suggested by the GA? The answer is given by the ready times of both jobs in Table 3. The sell department arranged a due date at *13.12.93* for N334. Later it came to be known that the needed raw material does not reach the workcenter before *2.2.94*. Thus it is obvious that N334 cannot be completed in time.

Job id	Due date	Event I		II	III	IV	V	VI	Just-in-Time	
		C_j	s						N_j	E_j
N333	0	9	x	0					9	
N320	127	131	x	0					4	
N334	0	171	x	0					171	
N321b	202	200		3					2	
N321a	202	238		2					36	
N322a	277	281	x	4					4	
N322b	277	287		0					10	
N323b	427	427	x	2					23	
N323a	427	405		0	3				69	
N324	495	497	x	0					72	
N377a	780			0					107	
N384	930			0	734	x	0		196	
N377c	780			1	773	x	0		7	
N377b	780			0	785		0			5
N378	855			0	855	x	0		*	
N397	915			0	871	x	0		44	
N391	1042			0	995	x	0		47	
N392	1095			0	1045	x	0		50	
N405	1080			0	1057	x	0		23	
N393	1185			3	1189	x	3			4
N394	1290			0	1272	x	0			6
N406	1350			2	1338	x	2	1296	x	4
N395	1365			3	1371	x	3	1382	x	6
N413	1455			2	1456		2	1459	3	17
N412	1635			0	1635	x	0	1639	x	1
N416	1672			2	1681	x	2	1703	x	4
N419	1785			0	1735	x	0	1752	x	31
N420	1785			0	1788	x	2	1788	x	32
Double shifts:										
		5	7	0	3	8	5	534	445	

Table 3. GA suggestion of scheduling on a rolling time basis.

To summarize, we count 534 hours of earliness and a slight smaller number of 445 hours of tardiness. These values are achieved by inserting a total of 28 double shifts and lead to the quantity $\overline{JIT} = 17.5$ in terms of our mean just-in-time measure. In comparison, the hand made schedule really implemented on *Collator 3* in 1994 is measured with a corresponding value of 33.3 by use of 38 double shifts.

5 Final Remarks

Within this paper we have developed an automated job scheduling system for an existing firm. The promising simulation results reveal the amount of unexploited machine capacity in the considered workcenter. There can be no doubt that this capacity has to be utilized in order to realize advanced concepts of production. In case of our workcenter the desired goal of just-in-time completion is approached by a good compromise between interdependent decisions which can be made by a GA. In other applications we will face different constellations of interacting dependencies but it appears to be likely that a GA will again successfully compromise.

From an academic point of view the constraints and the size of the application considered appears to be still moderate. Nevertheless, an expert could hardly improve the given solution. Thus it seem worthwhile to investigate the power of general-purpose optimization techniques like GAs even in the context of an apparently easy real-world problem. From the firms point of view the achieved results are so much encouraging that it intends to use the GA for machine scheduling in the near future.

References

- Bierwirth, C.; Kopfer, H.(1994): Dynamic Task Scheduling with Genetic Algorithm in Manufacturing Systems. University of Bremen, Chair of Logistics, Bremen
- Fry, T.D.; Armstrong, R.D.; Rosen L.D.(1990): Single Machine Scheduling to Minimize Mean Absolute Lateness: A heuristic solution. Computers Operational Research 17: 105–112
- Pesch, E.(1994): Learning in Automated Manufacturing. Physica Verlag, Heidelberg
- Raman, N.; Rachamadugu, R.; Talbot, F.B.(1989): Real-Time Scheduling of an Automated Manufacturing Center. European Journal of OR 40: 222–242
- Rixen, I.; Kopfer, H.(1994): Ein Genetischer Algorithmus für das Job Shop Scheduling Problem. University of Bremen, Chair of Logistics, Bremen
- Syswerda, G.(1991): Schedule Optimization using Genetic Algorithms. Davis, L.(ed.): Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York: 332–349
- Womack, J.P.; Jones, D.T.; Roos, D. (1990): The Machine that Changed the World. Macmillian Publishing Company, New York