Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls, eds.

GENERATING, BENCHMARKING AND SIMULATING PRODUCTION SCHEDULES - FROM FORMALISATION TO REAL PROBLEMS -

Gert Zülch
Kaiserstr. 12
ifab-Institute of Human and Industrial Engineering
University of Karlsruhe
Karlsruhe, 76128, GERMANY

Peter Steininger
Kaiserstr. 12
Steinbuch Centre for Computing (SCC), formerly at the ifab-Institute of Human and Industrial Engineering
University of Karlsruhe
Karlsruhe, 76128, GERMANY

Thilo Gamber
Michael Leupold
Kaiserstr. 12
ifab-Institute of Human and Industrial Engineering
University of Karlsruhe
Karlsruhe, 76128, GERMANY

ABSTRACT

Production scheduling has attracted the interest of production economics communities for decades, but there is still a gap between academic research, real-world problems, operations research and simulation. Genetic Algorithms (GA) represent a technique that has already been applied to a variety of combinatorial problems. Simulation can be used to find a solution to problems through repetitive simulation runs or to prove a solution computed by an optimization algorithm. We will explain the application of two special GAs for job-shop and resource-constrained project scheduling problems trying to bridge the gap between problem solving by algorithm and by simulation. Possible goals for scheduling problems are to minimize the makespan of a production program or to increase the due-date reliability of jobs or possibly any goal which can be described in a mathematical expression. The approach focuses on integrating a GA into a commercial software product and verifying the results with simulation.

1 PROBLEM DESCRIPTION

The resource-constrained project scheduling problem (RCPSP) is one of the classic scheduling problems of operations research. Already very simple special cases of the job-shop problem (JSP) which represents a special case of the RCPSP are NΠ-complete. An instance of JSP with ten jobs to be processed on ten resources (machines) formulated in 1963 remained open for more than 25 years. It was finally solved by a branch-and-bound algorithm.

We explain different computed solutions by two GAs in relation to simulation results. The GAs were developed to find a solution with the maximum results for a given set of production logistical objectives. The GAs and their operators are tested on benchmark instances and real-world data from a one-of-a-kind manufacturing department of a major German company. To have the possibility to work with real-world data we extended Microsoft Project with GA and a range of other aspects: A new graphical user interface is introduced to support users with a guided wizard describing the problem for which an optimal schedule is to be found. It includes different aggregation operators for the combination of objectives. Furthermore, the efficiency of the algorithm was compared to benchmark tests available in literature.

1.1 Characteristics of resource-constrained project scheduling problems

Many jobs in industry and elsewhere require a collection of activities to be completed while satisfying temporal, resource and precedence constraints. Temporal constraints demand some activities, or a set of them to be started or finished before or only after a certain point in time. Each activity uses an amount of renewable and non-renewable resources and is constrained by the fact that the total resource allocation per time-unit cannot exceed the total resource availability. Renewable resources include all resources of which a specified number of units is available for every period of the planning horizon (machinery, human resources) whereas non-renewable resources (such as material) are the ones of which only a fixed amount of units is available throughout the whole planning horizon. Precedence constraints refer to the technologically imposed winding-up of the operations (activities) within a production order (job). The objective is to create a schedule specifying when each operation within a production program is to begin (or finish) and what resources it will use in order to satisfy all constraints while pursuing an objective, e.g., taking as little total time as possible for all jobs (makespan), minimizing the mean tardiness of jobs, minimizing the maximum tardiness and so on.

The RCPSP belongs to the class of NΠ-complete optimization problems (Błażewicz, Lenstra, and Rinnooy Kan, 1983). This means that there is probably no efficient procedure for accurately finding the shortest schedule for arbitrary instances of this class of problems.

1.2 The job-shop scheduling problem

The JSP is an important and well-known special case of the RCPSP. A JSP consists of several sequences of operations so that each operation of a job has at most one successor and one predecessor. These jobs of a production program are connected by a common virtual starting and ending operation.

Furthermore, each resource type is available exactly once and each operation in a JSP needs at most one unit of a single resource type. This basic model may be extended by adding characteristics such as buffers, transportation, setup times, time lags, etc. (ref. chapter 1.4.2), allowing practical scheduling problems to be modeled more precisely.

1.3 Formal problem description

An instance of the RCPSP we are targeting consists of a set of *NOA* activities *i* and a set of *NOR* renewable resource types *n*. In literature there is also the notion of a set of non-renewable resource types which we are not considering. Resource availability *RAV* is specified using the tuple

$$RAV = (RAV_1, \ldots, RAV_n, \ldots, RAV_{NOR})$$

meaning RAV_n units of resource type *n* are available. Resource requirements RRA_i are specified per activity *i* using tuples

$$RRA_i = (RRA_{i,1}, \ldots, RRA_{i,n}, \ldots, RRA_{i,NOR})$$

where $RRA_{i,n}$ is the number of units of renewable resource type *n* required by activity *i*. In a JSP activities are aggregated into *NOJ* jobs or orders each of them comprising of activities with precedence constraints. The number of activities in job *j* is expressed by NAJ_j. Activity *i* of job *j* is noted using a tuple *(j,i)*.

The precedence constraints within job or order *j* describe the technological sequence (routing) of activities expressed by the set PRC_j consisting of relations $h \succ i$ denoting that activity *i* is a direct successor of activity *h*. An activity must be finished before each of its successors can be started. The technological sequence of job *j* can be expressed as:

$$PRC_j = \{(j,i) \succ (j,i+1) \mid i=1, \ \ldots, NAJ_j-1\}$$

meaning each activity *i* inside job *j* has activity *i+1* as its only successor. Furthermore, every resource type is available exactly once $(RAV_1, \ldots, RAV_{NOR}=1)$ and each activity requires exactly one unit of a single resource type during its entire execution. This means that two activities cannot be scheduled at the same time if they both require the same resource type.

To form a contiguous graph of all *NOJ* jobs within a production program virtual activities *a* and *e* called source and sink are added. All activities without a predecessor are made the source's successor, whereas all activities without a successor are made the sink's predecessor. Temporal constraints which require a job *j* to be started after a certain point in time are represented by virtual activities r_j which are inserted between the source *a* and job *j*. The scheduling problem of a production program can be represented by a graph as shown in Figure 1. In addition to the activity nodes and the source and sink nodes, it contains two dotted nodes called r_2 and r_3 which describe an imposed later start of jobs *2* and *3* relative to job *1*.

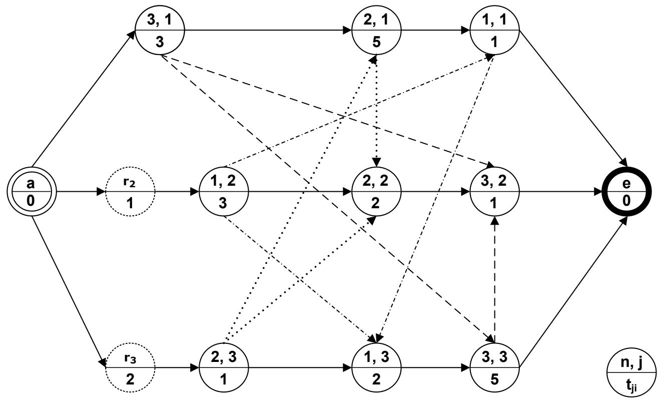


Figure 1: Network representation of a JSP (Steininger 2007).

In Figure 1, the directed arcs running from the source node *a*, through each activity node *(j,i)* to the sink *e* describe the technological sequence of activities within job *j* based on the routing PRC_j. Each node shows the job number *j*, the resource *n* needed and the operation time t_{ji}. There are also dotted directed arcs in the network linking all activities which require the same resource type and as such cannot be running in parallel. This kind of representation is called a disjunctive network.

1.4 Classification of scheduling problems

Classes of scheduling problems can be specified in terms of the three-field classification approach initially introduced by Conway, Maxwell, and Miller (1967, 2003). This classification was extended by Graham, Lawler, and Lenstra (1979), and later on by Brucker (2004). This three-field classification is described as $\alpha \mid \beta \mid \gamma$, where α specifies the machine environment (resources), β specifies the job and activity characteristics and γ describes the objective function or a combination of objective functions. Literature shows that this classification is still under continuous reconsideration.

Each symbol can be subdivided into more than one entry. If a symbol is subdivided, classification is separated with a comma and describes a more specific scheduling problem. It is also possible to leave every symbol but γ blank. A blank symbol $\circ$ identifies a default value.

1.4.1 Machine characteristic

The symbol α can be subdivided into α_1 and α_2. By subdividing α, the machine environment breaks up into different classes of scheduling problems. α_1 describes kind, order and count of machines. Brucker (2004) references 12 values. Additionally Brucker et al. (1999) specify machine characteristics for project scheduling, so that the following values are possible:

$$\alpha_1 \in \{\circ, P, Q, R, PMPM, QMPM, G, J, F, PF, O, X, FF, PS\}. \tag{1}$$

Based on these works we can formally specify terms for the possible values (ref. Table 11). α_2 specifies the number of machines (*NOR*) which could be defined as fixed value or left empty for all possible values, the default value $\circ$.

1.4.2 Job characteristics

Job characteristics are defined in many different details. Graham, Lawler, and Lenstra (1979) specify six possible values for β. Błażewicz et al. (2001) define eight possible values for β and Domschke, Scholl, and Voß (1997) subdivide β into ten possible values (ref. Table 22).

1.4.3 Objective function

Many objective functions are possible for scheduling problems. The three-field classification notes γ as the objective function or a combination of objective functions. As mentioned earlier the most pragmatic function is minimizing makespan (C_{max}), in other words the overall completion time of all jobs in a production program. For a selection of other possible objective functions, see Brucker (2004) and Steininger (2007).

1.5 Problem class specification of the problem under examination

Using the three-field classification to specify the problem instance of the JSP we are examining (ref. chapter 2), the following taxonomy is noted:

$$J \mid \circ \mid C_{max} \tag{2}$$

Formula (2) describes a class of scheduling problems as JSP (*J*) with a variable job and machine count. γ specifies the "traditional" objective function C_{max}, which describes our goal as taking as little makespan as possible for the schedule of all *NOA* activities using *NOR* machines.

According to Brucker et al. (1999), the RCPSP we are examining later (ref. chapter 4) can be specified using the following taxonomy:

$$PS \mid prec \mid C_{max} \tag{3}$$

Formula (3) describes a class of RCPSP with an unspecified number of machines, machine availability and machine requirements dependent on the actual input data. *prec* allows the job (order) to consist of a variable number of activities with precedence constraints forming an arbitrary acyclic graph.

Table 1: Machine characteristic α_1 (Steininger 2007; Brucker et al. 1999).

α_1	Terms for α_1			
$\circ$	$NOR = 1$	$\forall j \in J : t_{j1} = t_j$		
P	$NOR > 1$	$\forall j \in J, n \in N : t_{jn} = t_j$	$\lvert J_n \rvert < NOR$	$\exists n \in N : RAV_n > 1$
Q		$t_{jn} = \frac{t_j}{v_n}$		
R		$t_{jn} = \frac{t_j}{v_{jn}}$		
$PMPM$		$\forall j \in J, n \in N : t_{jn} = t_j$		
$QMPM$		$t_{jn} = \frac{t_j}{v_n}$		
G		$(j,0) \succ \ldots \succ (j,i) \succ \ldots \succ (j, NAJ_j)$	$NOR > 1$	$\forall n \in N : RAV_n = 1$
J			$NAJ_j > 1$	
F or PF			$\forall j,k \in J : NAJ_j = NAJ_k$, $RRA_j = RRA_k$, $PRC_j = PRC_k$	
O		$(j,0),\ldots,(j,i),\ldots,(j, NAJ_j)$		
X		$(j,0) \succ \ldots \succ (j,i) \succ \ldots \succ (j, NAJ_j)$		
FF			$NOR > 1$	$\exists n \in N : RAV_n > 1$
PS		$\exists g,h,i \in I : g \succ i, h \succ i$		$\exists n \in N : RAV_n > 1$

Legend:

J_n Set of jobs j requiring resource type n	N Set of resource types n	v_{jn} Processing speed of job j on resource type n
J Set of jobs j	t_{jn} Duration of job j on resource type n	
I Set of activities i	t_j Duration of job j	v_n Processing speed of resource type n

Table 2: Job characteristic β (cf. Domschke, Scholl, and Voß 1997).

β	β_1	β_2	β_3	β_4	β_5	β_6	β_7	β_8	β_9	β_{10}
Characteristic	Job count	Preemption of job	Precedence relations (routing)	Release dates	Activity duration (operation time)	Setup time	Resource restriction	Due date (specified)	Activity count	Buffer capacity

2 A GENETIC ALGORITHM FOR JOB-SHOP SCHEDULING

2.1 General principle

The term Genetic Algorithm (GA) describes a set of methods, which can be used to optimize complex problems. As the name suggests, the processes employed by GAs are inspired by natural selection and genetic variation. A GA uses a population of possible solutions to a problem and applies iterations in order to modify them. These iterations mimic those in nature in such a way that subsequent populations are fitter and more adapted to their environment compared to their predecessors. As generations progress over time, they become better suited to their environment and provide an advantageous solution in a given time.

Since their development (Holland 1975) GAs have been used to find solutions for many types of complex problems. A unique characteristic of a GA is that the fundamental algorithm is unaware of the problem it is optimizing. All that is required is that the parameters entered into the model can be efficiently transformed to and from a suitable GA chromosome format. Detailed introductions to GAs are given by Goldberg (1989) and Davis (1996).

The flowchart for the GA developed here is given in Figure 2. First, an initial population of randomly generated feasible schedules of the operations in the production program is created. These individual schedules form chromosomes which are subject to evolution. Once an initial population has been formed, selection, crossover, and mutation operations are performed repeatedly until the fittest member of the evolving population converges to a near-optimal fitness value. Alternatively, the GA may run for a user-defined number of iterations (Goldberg 1989).

The size of the population is user-defined and the fitness of each individual, in this case a schedule, is calculated according to a fitness function, in our case the makespan or an additive combination of different goal functions. Hence, it is also possible to use a fitness function on other calculated values like maximum tardiness, mean tardiness, number of tardy operations or jobs, etc. The schedules are then ranked according to the value of their fitness function and, after that, selected for reproduction.

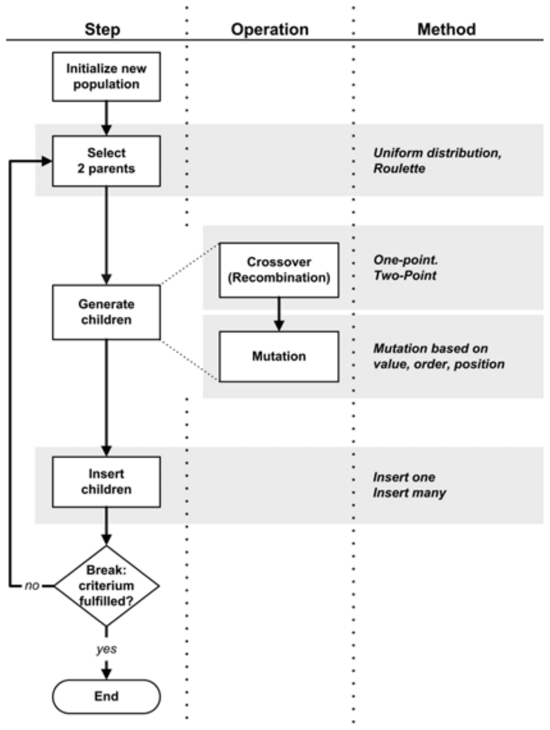

Step
Operation
Method
Initialize new population
Select 2 parents
Uniform distribution, Roulette
Crossover (Recombination)
One-point. Two-Point
Generate children
Mutation
Mutation based on value, order, position
Insert children
Insert one Insert many
Break: criterium fulfilled?
no
yes
End

Figure 2: Principal flow of a Genetic Algorithm (following Goldberg 1989).

2.2 Schedule encoding and decoding

GAs were derived from examining biological systems. In biological systems evolution takes place on chromosomes which are organic devices that program the structure of living beings. Following this line of argument, a living being is a decoded structure of all chromosomes. Natural selection is the link between chromosomes and the performance of the decoded structure. When implementing the GA, the variables that characterize an individual are represented in arrays (by index ordered lists). Each variable corresponds to a gene and the array corresponds to a chromosome in a biological system.

We decided to use the encoding scheme introduced by Fan, Ross, and Corne (1993) to build the chromosomes (Steininger 2007). Chen, Gen, and Yasuhiro (1996) call this scheme "operation based representation". Encoding starts by specifying jobs and corresponding operations in a list. Each operation in a job is encoded with the numerical id of the job in which it resides. Following that description, all jobs and activities are encoded in one potential schedule for the problem. The result is a chromosome representing a potential schedule.

2.3 Genetic operators

Genetic variation is indispensable for the progression of evolution. Genetic operators used in GAs are similar to those which occur in nature: asexual or sexual reproduction (crossover, also called recombination), mutation and survival of the fittest, or selection.

Crossover is the most delicate operation of GA. In our case, it may lead to the production of irregular operation sequences within a job. The GA developed here uses crossover, where mating chromosomes are cut once. We use a corrected two-point crossover to avoid non-regular operation sequences of orders, which Goldberg (1989) refers to as a PMX-crossover operator (for detailed reference and example see Steininger 2007).

Mutation is the process of randomly exchanging values of genes within a chromosome with small probability. It is not a primary operator, but it ensures that the possibility of searching any section in the problem space is never zero and prevents complete loss of genetic material through reproduction and crossover. We execute the mutation operator as a permutation by first picking (and deleting) a gene before reinserting it at a randomly chosen position of the chromosome (Steininger 2007).

To selectively reproduce the population and to determine the next generation we use a hit and miss selection procedure based on the fitness function. This could be implemented using a roulette wheel approach. An imaginary roulette wheel is constructed with a segment for each individual in the population. An individual's section size is based on the fitness value of the particular individual. A fit individual will occupy a larger slice of the roulette wheel than a weaker one (Steininger 2007,). Selection is made by rotating the roulette wheel a number of times equal to the population size. When the roulette wheel stops, the individual it points to is selected. This means that the more fit individuals will have a tendency to be selected more frequently than weaker ones.

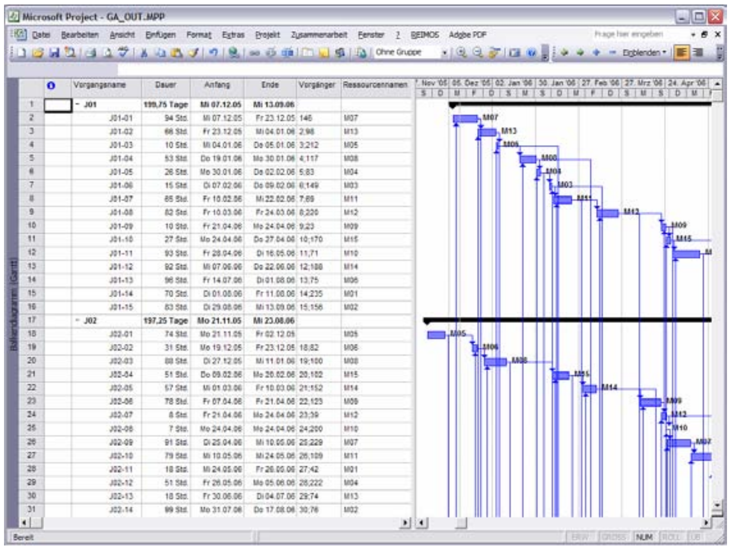
2.4 Fitness

The fitness function evaluates the fitness of each individual in the population, it depends on the specific application. Since a GA proceeds toward more fit individuals and the fitness value is the only information available to the GA, the performance of the algorithm is highly sensitive to the fitness function. In case of optimization routines, the fitness is the value of the objective function to be optimized (Steininger 2007).

3 IMPLEMENTATION AND PERFORMANCE TEST

3.1 Implementation

We implemented this scheduling algorithm in a software tool called REIMOS (German abbreviation for "Sequence planning for multi-product manufacturing systems"; Steininger 2007). In order to make use of existing industrial data, this software tool was integrated as an add-on into Microsoft Project 2003 (Figure 3).



	Vorgangsname	Dauer	Anfang	Ende	Vorgänger	Ressourcennamen
1	- J01	199,75 Tage	Mi 07.12.05	Mi 13.09.06		
2	J01-01	94 Std.	Mi 07.12.05	Fr 23.12.05	146	M07
3	J01-02	66 Std.	Fr 23.12.05	Mi 04.01.06	2;98	M13
4	J01-03	10 Std.	Mi 04.01.06	Do 05.01.06	3;212	M05
5	J01-04	53 Std.	Do 19.01.06	Mo 30.01.06	4;117	M08
6	J01-05	26 Std.	Mo 30.01.06	Do 02.02.06	5;83	M04
7	J01-06	15 Std.	Di 07.02.06	Do 09.02.06	6;149	M03
8	J01-07	65 Std.	Fr 10.02.06	Mi 22.02.06	7;69	M11
9	J01-08	82 Std.	Fr 10.03.06	Fr 24.03.06	8;220	M12
10	J01-09	10 Std.	Fr 21.04.06	Mo 24.04.06	9;23	M09
11	J01-10	27 Std.	Mo 24.04.06	Do 27.04.06	10;170	M15
12	J01-11	93 Std.	Fr 28.04.06	Di 16.05.06	11;71	M10
13	J01-12	92 Std.	Mi 07.06.06	Do 22.06.06	12;188	M14
14	J01-13	96 Std.	Fr 14.07.06	Di 01.08.06	13;75	M06
15	J01-14	70 Std.	Di 01.08.06	Fr 11.08.06	14;235	M01
16	J01-15	63 Std.	Di 29.08.06	Mi 13.09.06	15;156	M02
17	- J02	197,25 Tage	Mo 21.11.05	Mi 23.08.06		
18	J02-01	74 Std.	Mo 21.11.05	Fr 02.12.05		M05
19	J02-02	31 Std.	Mo 19.12.05	Fr 23.12.05	18;82	M06
20	J02-03	88 Std.	Di 27.12.05	Mi 11.01.06	19;100	M08
21	J02-04	51 Std.	Do 09.02.06	Mo 20.02.06	20;102	M15
22	J02-05	57 Std.	Mi 01.03.06	Fr 10.03.06	21;152	M14
23	J02-06	78 Std.	Fr 07.04.06	Fr 21.04.06	22;123	M09
24	J02-07	8 Std.	Fr 21.04.06	Mo 24.04.06	23;39	M12
25	J02-08	7 Std.	Mo 24.04.06	Mo 24.04.06	24;200	M10
26	J02-09	91 Std.	Di 25.04.06	Mi 10.05.06	25;229	M07
27	J02-10	79 Std.	Mi 10.05.06	Mi 24.05.06	26;109	M11
28	J02-11	18 Std.	Mi 24.05.06	Fr 26.05.06	27;42	M01
29	J02-12	51 Std.	Fr 26.05.06	Mo 05.06.06	28;222	M04
30	J02-13	18 Std.	Fr 30.06.06	Di 04.07.06	29;74	M13
31	J02-14	99 Std.	Mo 31.07.06	Do 17.08.06	30;76	M02

Figure 3: Microsoft Project 2003 showing the Gantt chart of a JSP optimized using REIMOS.

3.2 Benchmark test

Early on, research of scheduling problems started a competition on the "best schedule" of a specific problem. For that reason some researchers designed scheduling problems which were difficult to solve, as so called "benchmark instances". Some modern benchmark instances are distributed by Taillard (2008) and called JSP-15-15, JSP-20-15, and JSP-50-20. The name of the benchmark is derived from the scheduling of a specific problem, in this case of the underlying problem class $J \mid \circ \mid C_{max}$. JSP-50-20 stands for 50 jobs on 20 machines and so on, in this example $J, NOR=20 \mid NOJ=50, NOA=1000 \mid C_{max}$. This author also is publishing the list of best results for each scheduling problem instance, allowing our own results to be compared with those of other researchers.

We used the named three benchmark instances to model Microsoft Project 2003 files as input for REIMOS and started a GA run with the parameter values mentioned above. Our algorithm for benchmarking was modified to write a so-called "debugging output", which allows us to know specific values for the GA at every step. For example, these values are: number of iterations, schedule with the lowest makespan calculated so far and so on. We compared the best schedule found thus far with the best known schedule of Taillard (2008) and calculated a quality level of our GA-derived schedule as percentage of the best known.

For all three benchmark instances we attained a quality level of around 92 % of the best known solution calculated over 1,000 generations GA runtime. On a so-called standard PC, one calculation ran around 10 to 20 minutes, which corresponded to our time goal for planning a real-world schedule. We could have gone even further and let it run for hours or days, but we wanted to have a realistic industrial scenario. The problem with the best known solution by Taillard (2008) is that its runtime, implementation of algorithm, computer, etc. is not specified. This makes it hard to determine "how good was best" from an economical point of view.

4 ENHANCEMENT OF THE GENETIC ALGORITHM TO RCPSP

A subsequent project was dedicated to the enhancement of the concept developed by Steininger (2007) to enable solutions to RCPSPs. The main differences to be found consist in the different structure of the production program which, in contrast to JSPs, may consist of any directed, acyclic graphs, allows for several parallel resources for each resource type, and therefore offers the possibility to assign more than one resource to each activity (operation) which are required simultaneously.

4.1 Schedule encoding and decoding

As there is no direct correlation between the position of an operation in the chromosome and the technological sequence of the operations in the job for higher degrees of cross linking of a net graph, the encoding of chromosomes chosen according to Bean (1994) is floating-point and priority based. Each activity is given a random priority represented by a floating point number between 0 and 1. Given the fact that the actual planning sequence of activities is not immediately apparent from this representation of chromosomes, it needs to be set up during a decoding phase prior to the use of the fitness function. In the literature two Schedule Generation Schemes (SGS) are known with regards to this matter, i.e. the serial and the parallel SGS (Kolisch and Hartmann 1999).

Taking a gradual approach, the serial SGS assigns the earliest possible starting point to each unscheduled activity without violating precedence and resource constraints. The "scheduled set" *SST* includes all initialized operations, whereas the "eligible set" *EST* includes all operations which can be initialized in the following step as all their predecessors are already in *SST*. During each step the activity with the highest priority is chosen from *EST*. Once the earliest possible starting point has been determined for this activity, it is deleted from *EST* and added to *SST*. Each of its successors – which now can be planned as all its immediate predecessors have already been initialized – is then added to *EST*. The process begins with $EST=\{source\}$, $SST=\emptyset$ and terminates as soon as the end activity has been planned and $EST=\emptyset$.

The parallel SGS, in turn, uses a method similar to discrete event simulation. Each planning step is determined by a planning point in time *PPT* and the "completed set" *CST*, "running set" *RST* and *EST* sets of activities completed, running and ready to be initialized at this point in time. The process starts with $CST=\emptyset$, $RST=\emptyset$, $EST=\{source\}$ and $PPT=0$. At the beginning of each planning step, all activities completed at time *PPT* are deleted from *RST* and added to *CST*; in addition, all successor activities whose predecessors are now completed are added to the eligible set *EST*. Subsequently, as many activities as possible are deleted from *EST*, initialized at point *PPT* and added to *RST* under the condition of not violating any resource constraints. High-priority activities take precedence over those rated with lower priorities. If no other activity from *EST* can be initialized at the current planning point, the next planning point *PPT* will be calculated as the earliest possible time an operation from the running set *RST* will be completed. The process terminates with planning of the sink activity.

Both SGSs only identify admissible planning solutions. Kolisch (1996) shows that the solution space identified by a parallel SGS is nothing but a sub-space contained in that of a serial SGS, and consequently, depending on the objective, does not necessarily include the optimal solution. Therefore a serial SGS was chosen for our GA implementation. The method thus corresponds to the Priority Value Based Genetic Algorithm developed by Hartmann (1998).

4.2 Genetic operators

As the priorities are selected randomly with the initiation of the GA, there is only a limited number of priority conflicts. In case there are two or more operations of the same priority in the eligible set *EST* at the same time, this conflict will be solved by having one of the operations selected randomly. This approach simplifies the use of genetic operators.

Mutation can be implemented by replacing one gene of the chromosome with a newly generated random priority. By solving conflicts in the above-described manner, the correction phase following crossover in the formerly used GA is no longer required, and a simple two-point crossover is used.

4.3 Comparison with benchmarks

A benchmark test with the RCPSP instances of the Project Scheduling Problem Library (PSPLIB; Kolisch and Sprecher 1996) was performed in order to verify the algorithm. These automatically generated instances (Kolisch, Sprecher, and Drexl 1995) represent the standard for comparison of algortihms solving RCPSPs. The instances consist of three sets and include either 30, 60 or 120 activities each of which could be interpreted as an operation inside a job. For each of the instances the 4 resource types and their availability varies. The problem sets with 30 and 60 operations consist of 480 problem instances, whereas those with 120 operations have 600. Thus using the classification introduced by Brucker et al. (1998) a problem instance of the set with 30 operations could be classified as $PS,4 \mid prec, NOA=30 \mid C_{max}$. For each of the instances, 10 runs of the enhanced REIMOS procedure were performed and the mean of makespans of all individual solutions was computed.

The results can be seen from Table 3. For the problem set with 30 operations each, the average deviation describes the standard deviation of the mean solution calculated with the enhanced REIMOS procedure compared to the optimal solutions of the benchmark instances, whereas for all other problem sets it describes the standard deviation from the lower bounds of the critical path of the production programs, as their optimal solutions are not known. The values in the column "best known" are calculated in the same fashion but show the deviation of the makespans calculated by the currently best known algorithms for each of the problem sets (cf. Kolisch and Hartmann 2006).

When comparing the above-described enhanced REIMOS procedure with the other algorithms presented by Kolisch and Hartmann (2006), it only falls within the lower midrange in terms of benchmark results. It must be taken into consideration, however, that the development of this procedure was not geared towards the solution of theoretical benchmarks but was conceived with real-world problems in mind. Along the same lines it is worth mentioning that in real life the volume of the data material is considerably larger and accompanied by the inclusion of large data sets.

Table 3: Best benchmark results for problem sets with 30, 60 and 120 operations (following Kolisch and Hartmann 2006).

Number of operations *NOA*	**Average deviation of makespans (in %)**	
	Enhanced REIMOS	Best known
30	0.50	0.00
60	12.94	10.71
120	41.36	31.24

4.4 Practical example

In real-life situations, additional constraints often need to be added to the list of temporal, precedence and resource constraints already mentioned. When planning the sequence of orders (jobs), for example, the work shifts of the operators and the available operating times of the machines need to be considered in more detail, which further restrict the availability of resources. If the interaction between operators and machines is taken into account, the problem becomes even more complex. In addition, there are often relations between the orders of a production program which go beyond the theoretical end-to-start relation without time lags. For this reason the SGS of the enhanced GA has been designed to additionally consider start-to-start, start-to-end and end-to-end relations with optional minimum time lags. This problem class can be noted as follows according to Brucker et al. (1999):

$$PS \mid prec, temp, r_j \mid C_{max}$$

In this taxonomy, *temp* notes any random start-to-start time lags, whereas r_j describes the possibility to select different points of initialization for incoming orders.

The enhanced REIMOS procedure was tested with data from the production program of a one-of-a-kind production at a major German company. The sequence problem under examination included 41 different products with a total of approx. 4,500 individual operations. The production system consisted of 30 different resource types and, taking into consideration parallel resources, a total of 41 machines and work stations. For confidentiality reasons, the model mix, job and operation data are under a non-disclosure agreement and therefore cannot be published here.

5 SIMULATION STUDY AND COMPARISON OF RESULTS

The following simulation study was aiming at the comparison of the results gained by the enhanced REIMOS procedure and those from a series of simulation runs. For the latter the organization-oriented simulation procedure FEMOS was used. FEMOS (German acronym for "Manufacturing and assembly simulator"; see for details Zülch and Grobel 1996; Zülch and Brinkmeier 1998) has been developed at the ifab-Institute of Human and Industrial Engineering of the University of Karlsruhe (Germany) and has proved itself in many transfer projects. The FEMOS procedure has been used to analyze not only organizational changes in the manufacturing area and in areas preceding manufacturing but also organizational forms along the entire production-logistical process chain have been investigated (cf. Zülch and Greinke 2004; Zülch and Warrisch 2004; Zülch 2008).

5.1 Data base and initialization policies

The data for this comparison is based on empirical values. The influencing factors, which are to change in the course of the simulation study, include the planned starting date (point of entry into the production system), the planned finish date and the sequence of production orders or orders of a product group, respectively, plus different priority rules. The term product group refers to a combination of individual products with similar work content. The priority rules comprise a selection of rules available within the FEMOS simulator: first come first served, shortest operation time, longest operation time, slack time, and earliest due date.

Furthermore, 8 order initialization policies were looked into. Initialization policy P1: sequence of order initialization as specified by experts of the industrial partner; P2: initialization according to product groups with increasing work content (the higher the work content the later the point of initialization), individual products from the product group initialized equidistantly; P3: initialization of individual products in equidistant intervals, product groups mixed; P4: product groups initialized in equidistant intervals, the lower the work content the later the point of initialization, individual products initialized equidistantly; P5: product groups initialized one after the other, the higher the work content the later the point of initialization, individual products initialized at the same point in time; P6: initialization of product groups in equal intervals, product groups mixed; P7: product groups initialized one after the other, the lower the work content the later the point of initialization, individual products initialized at the same point in time; P8: block initialization of all products at point zero.

According to this test plan, individual order initialization policies were simulated first, followed by the simulation of order initialization policies in combination with priority rules. The second test plan used the mean value analysis to identify those resources constituting bottleneck resources due to their high utilization. They were then examined by means of several specific order initialization policies by choosing processing- and arrival time-based priority rules for them as opposed to the date-based priority rules used for all other resources. In the third test plan, the most important operations of the conclusive assembly processes were given higher priorities than the preceding assembly operations. This means that the operations rated with high priority were those at the end of the manufacturing process which would – according to the hypothesis – prove most effective with regards to the reduction of makespan and an increase in reliability of delivery.

5.2 Comparison of results achieved with the enhanced REIMOS procedure and with the simulator FEMOS

The FEMOS simulation runs were evaluated based on the target parameters makespan, delivery reliability and resource utilization. For the comparison the lowest makespans of the REIMOS procedure and those of the FEMOS simulator achieved with order initialization policies P1 and P8 were used. The reasons for choosing these policies for further analyses were that policy 1 was the best policy identified by the industrial partner up to that point and that policy P8 allowed for the highest possible degree of flexibility as all orders are initialized at point zero. This ensures a solid basis for the comparison of results from both procedures. Out of the two initialization policies, those with the best result in combination with the priority rules were selected, namely shortest operation time (SOT) and longest operation time (LOT). In addition, the simulation results were compared with regards to the respective makespan of the product groups (block makespan). Figure 4 sets the makespans of the sequences identified by means of the REIMOS planning method alongside those gained by the FEMOS simulation runs. The strategies used were the priority rule SOT in combination with order initialization policy P1 and LOT in combination with order initialization policy P8 respectively.

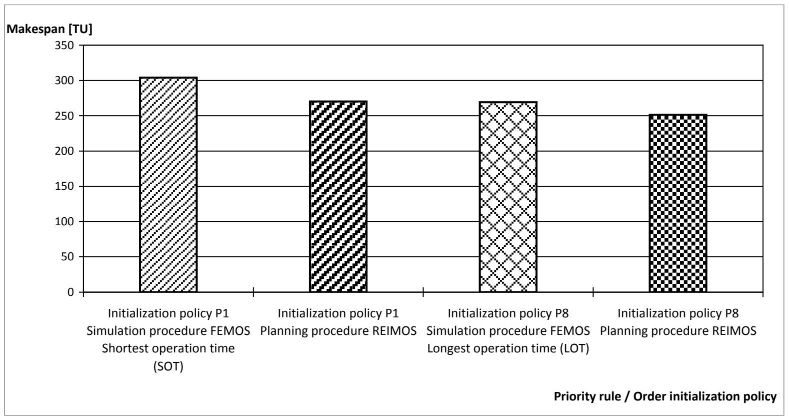


Figure 4: Comparison of makespans identified by means of FEMOS using order initialization policies and priority rules and those generated with the enhanced REIMOS planning procedure.

Within the simulation study, the comparison of makespans gave the best result for order initialization policy P1 in combination with the SOT rule, whereas order initialization policy P8 proved to be most effective when combined with the LOT rule.

The comparison with the FEMOS results shows that REIMOS gives better results for both order initialization policies. The makespan achieved with REIMOS is approx. 11 % lower for order initialization policy P1 and almost 7 % lower for policy P8 compared with sequence planning on the basis of priority rules.

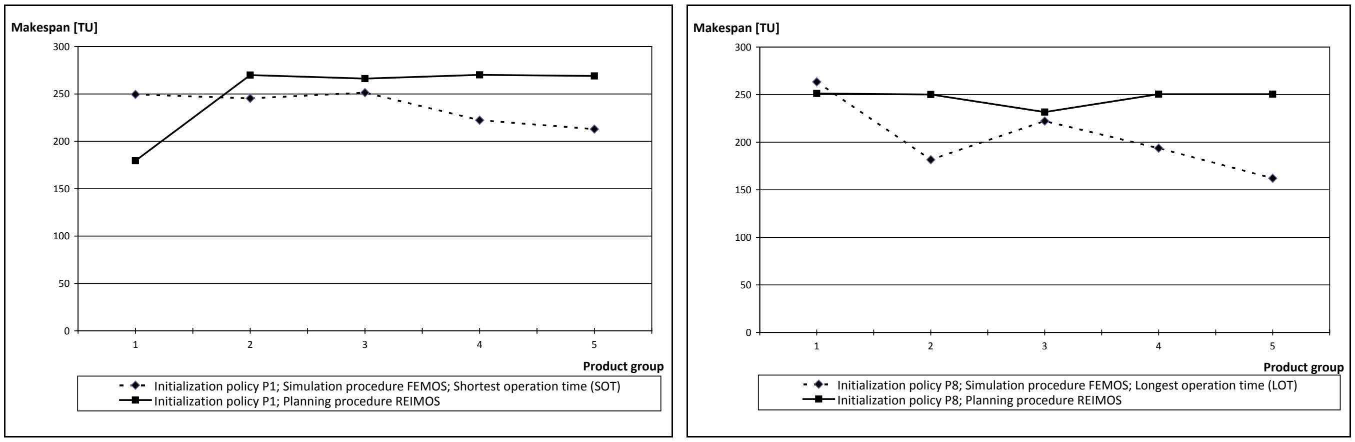


Figure 5: Comparison of makespans of production groups when using simulation with priority rules and when using the enhanced REIMOS planning procedure.

The diagrams illustrated in Figure 5 are the product of a comparison of makespans for production orders of the 5 product groups generated with the REIMOS planning procedure and the simulation of priority rules with FEMOS respectively. When the makespans of individual product groups for order initialization policy P1 are compared it turns out that the values are 6 % higher with REIMOS than when the priority rules using FEMOS were applied. A comparison of the makespans of the product groups for order initialization policy P8 delivers the similar key figures: The values of the REIMOS planning procedure are approx. 21 % higher than those generated with the simulated use of priority rules. The individual makespans of orders within one product group do not vary strongly from the group's mean value. This gives rise to assumptions as to the positive correlation between an equidistant or almost equidistant spacing of inter-arrival times within a product group and an improvement of makespans.

All of the above proves that the makespan results delivered by the use of simulation with priority rules fall short of those delivered by the Genetic Algorithm within the enhanced REIMOS planning procedure. This proves that the more complex Genetic Algorithm is superior to the priority rule-based simulation approach.

6 CONCLUSIONS AND FURTHER DEVELOPMENTS

A comparison of the best FEMOS results of the simulation study at hand with the best results of REIMOS shows that the more complex REIMOS planning procedure generates lower makespans than the use of priority rules in the simulation runs. But the comparison also proved that the individual makespans of the product groups show higher values for REIMOS than when using FEMOS with priority rules. Depending on the target of optimization, the approaches lead to diverging conclusions. As far as multi-target optimization tasks are concerned, simulation therefore represents a means to verify the results with respect to the weighting factors of the objective function.

When using heuristics and suboptimal methods such as GAs, it is often crucial to implement these criteria separately. These enhancements can be facilitated by combining simulation and optimization procedures from operations research. One concrete scenario would be the use of the simulation tool as a fitness function of the GA in order to allow for more complex weighting functions to be taken into account.

It should also be noted that disturbances, decisions, material flow, etc., can be taken into consideration when the values of a fitness function are calculated from simulation runs. The next step would be to focus on another reduction of the calculating time of such a method. One possibility might be to distribute the calculation of the fitness functions to different processor cores or to different computers (grid computing). All these ideas are reason to engage in further research in this field.

REFERENCES

Bean, J. C. 1994. Genetic Algorithms and Random Keys for Sequencing and Optimizations. *ORSA Journal of Computing* 6:154-160.

Błażewicz, J., K. H. Eecker, E. Pesch, G. Schmidt, and J. Węglarz. 2001. *Scheduling Computer and manufacturing processes*. Berlin, Heidelberg, New York et al.: Springer Verlag, 2nd ed.

Błażewicz, J., J. Lenstra, and A. H. G. Rinnooy Kan. 1983. Scheduling subject to resource constraints: Classification and complexity. *Discrete Applied Mathematics* 5:11-24.

Brucker, P. 2004. *Scheduling Algorithms*. Berlin, Heidelberg, New York et al.: Springer Verlag.

Brucker, P., A. Drexl, R. Möhring, K. Neumann, and E. Pesch. 1999. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research* 112:3-41.

Cheng, R., M. Gen, and Y. Tsujimura. 1996. A tutorial survey of job-shop scheduling problems using genetic algorithms – I. representation. *Computers & Industrial Engineering* 30:983-997.

Cherkassky, B. V., A. V. Goldberg, and T. Radzik. 1993. Shortest Paths Algorithms: Theory and Experimental Evaluation. Technical Report 93-1480. Stanford: Stanford University, Computer Science Department.

Conway, R. W., W. L. Maxwell, and L. W. Miller. 1967. *Theory of Scheduling.* Reading (MA): Addison-Wesley Publishing Company.

Conway, R. W., W. L. Maxwell, and L. W. Miller. 2003. *Theory of Scheduling*. Mineola (NY): Dover Publications.

Davis, L. 1996. *Handbook of Genetic Algorithms*. Florence (KY): International Thomson Computer Press.

Domschke, W., A. Scholl, and S. Voß. *Produktionsplanung*. Berlin, Heidelberg, New York et al.: Springer Verlag, 2nd ed. 1997.

Fan, H.-L., P. Ross, and D. Corne. 1993. A Promising Genetic Algorithm Approach to Job-Shop Scheduling, Rescheduling, and Open-Shop Scheduling Problems. In *Proceedings of the Fifth International Conference on Genetic Algorithms*, ed. S. Forrest, 375-382. San Mateo (CA): Morgan Kaufmann.

Gallo, G., and S. Pallottino. 1982. A new Algorithm to find the Shortest Paths between all Pairs of Nodes. *Discrete Applied Mathematics* 3 (4), 23-25.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston (MA), München: Addison Wesley.

Graham, R. L., E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan. 1979. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics* 16:287-326.

Hartmann, S. 1998. A Competitive Genetic Algorithm for Resource-Constrained Project Scheduling. *Naval Research Logistics* 45:733-750.

Holland, H. J. 1975. *Adaptation in natural and artificial systems*. Cambridge (MA), Ann Arbor (MI): University of Michigan Press.

Kolisch, R. 1996. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research* 90:320-333.

Kolisch, R., and S. Hartmann. 1999. Heuristic algorithms for the resource-constrained project scheduling problem: Classification and computational analysis. In *Project Scheduling – Recent Models, Algorithms and Applications*, ed. J. Węglarz, 147-178. Boston (MA): Kluwer Academic Publishers.

Kolisch, R., and S. Hartmann. 2006. Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research* 174:23-37.

Kolisch, R., and A. Sprecher. 1996. PSPLIB – A project scheduling problem library. *European Journal of Operational Research* 96:205-216.

Kolisch, R., A. Sprecher, and A. Drexl. 1995. Characterization and Generation of a General Class of Resource-Constrained Project Scheduling Problems. *Management Science* 41:1693-1703.

Steininger, P. 2007. *Eine Methode zur Reihenfolgeplanung bei Mehrprodukt-Fertigungssystemen*. Dissertation, University of Karlsruhe. Aachen: Shaker Verlag.

Taillard, É. D. 2006. Scheduling instances. Available via <http://ina2.eivd.ch/Collaborateurs/etd/problemes.dir/ordonnancement.dir/ordonnancement.html> [accessed December 20, 2008].

Zülch, G. 2008. Schnelle Fertigungsanläufe als Beitrag zur Beschäftigungssicherung. In *Beiträge der Arbeits- und Betriebsorganisation zur Beschäftigungssicherung*, ed. G. Zülch, 47-64. Wiesbaden: Gabler GWW Fachverlage.

Zülch, G., and B. Brinkmeier. 1998. Simulation of Activity Costs for the Reengineering of Production Systems. *International Journal of Production Economics, Special Issue* 56/57, 711-722.

Zülch, G., and J. Greinke. 2004. Simulation-aided Reconfiguration of an Industrial Service System for the Repair of Electrical Tools. Espoo: Sim-Serv. Available via <http://www.sim-serv.com/pdf/whitepapers/whitepaper_73.pdf> [accessed March 19, 2009].

Zülch, G., and T. Grobel. 1996. Shaping the organization of order processing with the simulation tool FEMOS. *International Journal of Production Economics, Special Issue* 46/47, 251-260.

Zülch, G., and W. Warrisch. 2004. Simulation-aided Segmentation of a Mechanical Parts Manufacturing. Espoo: Sim-Serv. Available via <http://www.sim-serv.com/pdf/whitepapers/whitepaper_74.pdf> [accessed March 19, 2009].

AUTHOR BIOGRAPHIES

GERT ZÜLCH Prof. Dr.-Ing. Dipl.-Wirtsch.-Ing., studied mechanical engineering at the Technical University of Braunschweig as well as industrial engineering at the Technical University of Aachen (Germany). Upon concluding his work at an industry-oriented research institute, he was employed in the Central Department for Manufacturing Economics and Organization at the Siemens headquarters in Munich. Since 1985 he has been head of the ifab-Institute of Human and Industrial Engineering of the University of Karlsruhe, which he founded. He is a member of a number of international organizations, both in the field of production management and the field of ergonomics. Furthermore, he has on many occasions acted as a scientific expert, both nationally and abroad (European Commission, President of the French Republic, Academy of Finland, German Science Council, German Research Association, Northrhine-Westfalian Academy of Sciences etc.). Since March 2009 he has chaired the German speaking Association of Ergonomics (Gesellschaft für Arbeitswissenschaft). His scientific interests range from communication ergonomics to the simulation of production systems with special attention to personnel assignment problems. <gert.zuelch@ifab.uni-karlsruhe.de>.

PETER STEININGER Dr.-Ing., studied industrial engineering with business studies at the University of Karlsruhe. He has worked in different software and consulting companies and is now a researcher at Steinbuch Centre for Computing (SCC) at the Karlsruhe Institute of Technology (KIT), which represents the merger of the University of Karlsruhe with the Research Center Karlsruhe. He is a member of a number of international organizations, both in the field of information technology and the field of production management and lecturer for database systems and information systems. His scientific interests range from information technology to production economics. <peter.steininger@KIT.edu>.

THILO GAMBER studied information engineering and management, cultural and general studies and industrial education at the University of Karlsruhe. He graduated in 2005 with a diploma degree in information engineering and management. Since graduation, he has worked for the ifab-Institute of Human and Industrial Engineering of the University of Karlsruhe as a research assistant. <thilo.gamber@ifab.uni-karlsruhe.de>.

MICHAEL LEUPOLD studies computer science at the University of Karlsruhe. Since 2006 he has worked for the ifab-Institute of Human and Industrial Engineering as part of the project staff. <michael.leupold@ifab.uni-karlsruhe.de>.