BENCHMARKING REAL-WORLD JOB-SHOP SCHEDULING USING A GENETIC ALGORITHM WITH A SIMULATION APPROACH

Dr.-Ing. Peter Steininger

Steinbuch Centre for Computing (SCC)
Karlsruhe Institute of Technology (KIT)

steininger@kit.edu

ABSTRACT

Although production scheduling has attracted the research interest of production economics communities for decades, a gap still remains between academic examples and real-world problems. Genetic Algorithms (GA) constitute techniques which have already been applied to a variety of combinatorial problems. I intend to explain the application of a GA approach to bridge this gap for job-shop scheduling problems, by minimizing the makespan of a production program or increasing the due-date reliability of jobs.

Simulation is a useful tool in problem solving. Here repetitive runs of simulated models or computed solutions through algorithms are applied. For job-shop and resource-constrained project scheduling, problems trying to bridge the gap between computed solutions and the feasibility of the simulation occur. I would like to explain the application of this special GA for job-shop and resource-constrained project scheduling. Possible goals for scheduling problems include minimizing the time required of a production program, or increasing the due-date reliability of jobs, or possibly other objectives which can be described in a mathematical expression. The approach focuses on integrating a GA into a commercial software product, namely Microsoft Project 2003, and verifying the results with the simulation.

Keywords: Job-shop scheduling, Genetic algorithms, Job-shop Benchmarks, Real-world scheduling problems, Simulation.

1. INTRODUCTION

1.1. Characteristics of job-shop scheduling problems

Many jobs in industry and elsewhere require a collection of tasks or activities to be completed whilst satisfying temporal, resource and precedence constraints. Temporal constraints refer to the time requirement that some tasks or set of tasks have, where they must be finished before or after a certain point in time. Resource constraints dictate that two tasks requiring the same resource cannot be carried out simultaneously. Whereas Precedence constraints refers to the technologically imposed carrying out of the tasks within a job or production order. The objective is to create a schedule specifying when each task or activity is to commence (or finish) and what resources it shall use in order to satisfy all the constraints while pursuing an objective. The overall goal is to complete all tasks to an acceptable standard within the least time possible (makespan), whilst taking into consideration such aspects as minimizing the mean tardiness and number of jobs. This is also referred to as the job-shop scheduling problem (JSP).

The JSP is a very important and well defined scheduling problem. It is a basic model, which may be extended through use of additional characteristics like buffers, transportation, setup time, time lags, etc., allowing practical scheduling problems to be modeled more precisely. In its general form, it is NP-complete, meaning that there is probably no efficient procedure for exactly finding the shortest schedule for arbitrary instances within this class of problem.

BAGCHI (p. 109) references the JSP as follows: "Within the great variety of production scheduling problems that exist, the job shop scheduling problem is one that has generated the largest number of studies. It has also earned a reputation for being notoriously difficult to solve. Nevertheless, the JSP illustrates at least some of the demands imposed by a wide array of real world scheduling problems… Attempts to tackle the multi objective job shop are still relatively few."

A JSP is usually solved using a heuristic algorithm which takes advantage of special properties of a specific class of instances. This can be regarded as a loophole to reduce the complexity of a given problem.

1.2. Formal problem description

An instance of the JSP consists of a set of NOA_i activities within jobs i and NOM machines j. Each job consists of a number of activities so that we may count the total number of activities NOA as follows:

$$NOA = \sum_{i=1}^{NOM} NOJ_i \qquad (1)$$

Here each job has a fixed number and a sequence of activities. Each activity requires a certain amount of time implementing a single machine for its entire duration. An activity must be finished before each activity following it can commence, with each job utilizing a different machine. Two activities cannot be scheduled at the same time if they both require the same machine. In all we need to find a feasible schedule

which minimizes some objective function, whilst reducing makespan. Where makespan expresses the overall completion time of all activities, see STEININGER (2007, pp. 26). This produces a complexity function for the JSP expressed as

$$O\left(\left(NOJ!\right)^{NOM}\right) \tag{2}$$

In order to find the best schedule for a problem instance, we could enumerate and evaluate all possible schedules. The number of feasible schedules to be enumerated would be the result of function (2).

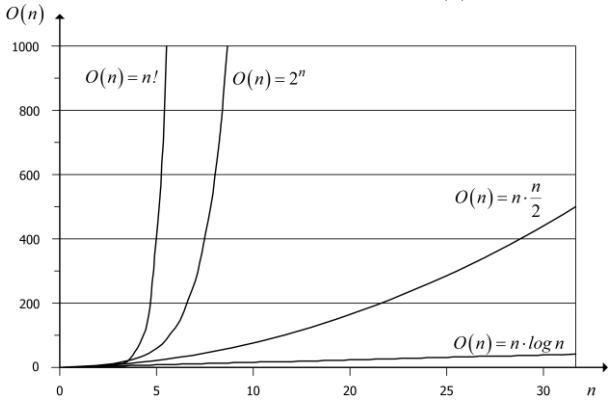

Figure 1. Selected complexity functions (STEININGER, p. 82).

Figure 1 illustrates the dimensions of selected complexity functions, where n represents the number of problem elements, which are determined here by the number of activities and machines. These graphs illustrate how the complexity of a JSP can be much larger than some other well-known problems, such as "Permutation problem" which is $O(n) = n!$, "Towers of Hanoi" which is $O(n) = 2^n$, and "Quicksort" which is $O(n) = n \cdot \log n$.

1.3. 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 (2003) and extended by GRAHAM (1979) and BRUCKER (2004). Of course this depicts a continuous developmental process open to scrutiny and reevaluation. This three-field classification is described as α | β | γ, where α specifies the machine environment, β specifies the job characteristics and γ represents either the objective function or a combination of objective functions. Using the three-field classification to specify the problem instance of the JSP we are examining, the following taxonomy is noted:

$$J, NOM \mid NOJ, \text{intree}, t_{ij} \mid C_{\max} \tag{3}$$

Formula (3) describes a class of scheduling problems as JSP (J) with a fixed machine count of NOM and a predefined and fixed number of NOA activities. The precedence routing of activities in each order is predefined and fixed as a directed graph with operation times (t_{ij}). These are expressed as integer values for each activity.

The three-field classification denotes γ as the objective function or a combination of objective functions. In formula (3) γ specifies the "traditional" objective function (C_{max}). This depicts our primary objective which is to limit the makespan using the least time possible for the schedule of all NOA activities using NOM machines.

2. MODELLING OF JSP SCHEDULING PROBLEMS

2.1. Formal problem representation

Even slightly different job-shop problems require completely different encodings in order to find a good solution. Thus, choosing an efficient representation is a very important component when solving a JSP. However, deciding upon a relevant representation for a scheduling problem is as difficult as choosing a good search algorithm for a decision problem. Not all algorithms work equally efficient in a specific problem representation. In order to describe the representation technique developed for our solution, a simple job-shop scheduling problem example has been used in Table 1.

Table 1. Example of a production schedule problem with 3 jobs, routing information S_j for the jobs i, 3 machines j and operation times t_{ij} for each task of a job in time units (TU).

Job j	**Machine** / S_j	t_{ij} [TU]		
		1 i	**2**	**3**
1	(3,2,1)	3	5	1
2	(1,2,3)	3	2	1
3	(2,1,3)	1	2	5

The scheduling problem can be represented by a graph as shown in Figure 2. In addition to the activity nodes *(i,j)*, it contains a source node a, and a sink node e, both with no durations (operation times), and two dotted nodes called r_2 and r_3 which describe a later start of jobs 2 and 3 imposed relative to job 1.

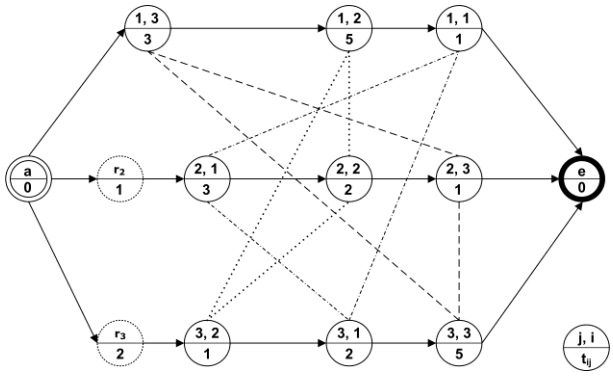

Figure 2. Network representation of a JSP based on Table 1 (STEININGER, p. 64).

The directed arcs running from the source node a, through each activity node (i, j) to the sink node e describes the technological sequence of activities based on the routing S_j in Table 1. Each node shows the job number i, the machine needed j and the operation time t_{ij}. There are also undirected arcs in the network, which reference all possible sequences of an activity in a given job on a specific machine. Such a representation is termed a disjunctive network.

2.2. Data representation and problem reduction

Care must be taken when adopting such a graphical representation into a data structure, especially for the JSP in an area with hundreds of activities, thousands of sequences and millions of possible actions carried out with specific machines.

A data structure which is very efficient in the use of storage (due to the size of a practical problem) as well as in time, can also be considered as depicting a network. GALLO and PALLOTTINO (1982) introduced the so-called "Forward Star" data structure, which is the most efficient portrayal of all existing network data structures for representing a network. The "Forward Star" data structure uses three arrays to describe a (directed) network. First we have an array termed $from$, whose index represents all nodes of the network. The value of an index field references the index of the second array named $succ$, which is the index of a node to connect, referencing $from$. The third array is labeled $cost$ and reports the cost of a specific arc connecting two nodes.

The "Forward Star" data structure allows for a perfect implementation of the activity order of any JSP. An efficient implementation in storage and time and a reduction of the initial problem described by the $\alpha \mid \beta \mid \gamma$ three-field classification is achieved/possible/occurs. Using the "Forward Star" data structure our problem is reduced to the following taxonomy (4):

$$J, NOM \mid NOJ, chains, t_{ij} \mid C_{\max} \qquad (4)$$

Care must also be taken when adopting the representational scheme and the associated operators for an effective algorithm. Here with the application of traditional problem solving with GAs , the chromosomes are implemented as binary vectors. Such an algorithm is an excellent choice for problems in which a point naturally maps into a chromosome of zeros and ones. Unfortunately, this approach of zeros and ones cannot be implemented for real-world engineering problems such as JSP, because of the amount of information needed to represent coding of the JSP. Therefore, we have to find a way to integrate the "Forward Star" data structure into a GA.

3. GENETIC ALGORITHMS

3.1. General principle

The term Genetic Algorithm 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. Thus GA uses a variety of possible solutions to a problem and applies repetition 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 in the late 1980's GAs have been used to find solutions to many types of 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 a model can be efficiently transformed to and from a suitable GA chromosome format. Therefore GA optimization can be applied to many types of complex problems.

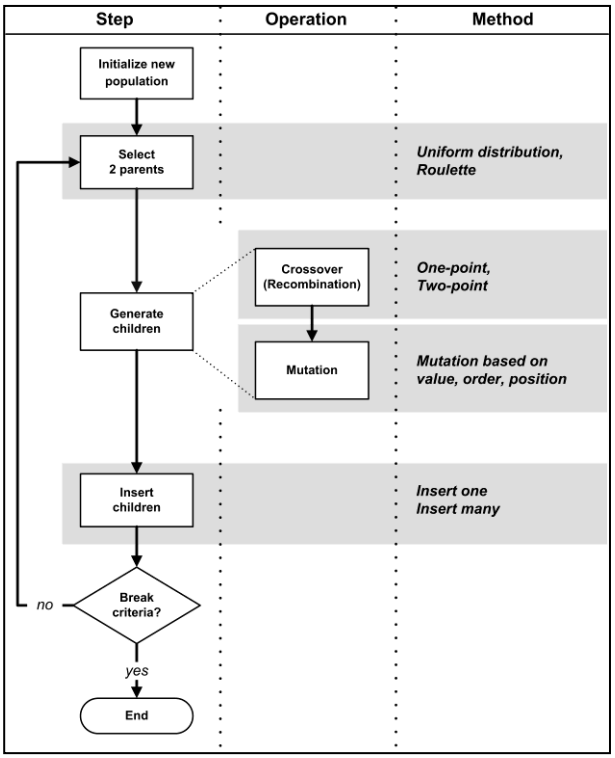

Figure 3. Principal flow of a Genetic Algorithm (following GOLDBERG (1989, pp.59)).

The flow-chart stages of the GA are as follows: First, an initial population of randomly generated sequences of the activities in the schedule 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.

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 this is the makespan or an additive combination of different

goals. It is also possible to use a fitness function on other calculated values like mean tardiness, maximum tardiness, number of tardy jobs and so on. Combinations of different functions in one fitness function are also possible. The schedules are then ranked according to the value of their fitness function and, after that, selected for reproduction.

3.2. Schedule encoding and decoding

GAs were derived from examining biological systems. In biological systems evolution takes place on chromosomes which are organic devices for programming the structure of living beings. In this sense, 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 is equivalent to a chromosome in a biological system.

We have decided to use the encoding schema introduced by BEAN (1994) to build the chromosomes termed "Operation Based Representation". Encoding commences with the enumeration of jobs and corresponding activities in a list. Each activity in a job is encoded with the numerical id of the job in which it resides. All jobs and activities are encoded following that description in one potential schedule for the problem. The result is a chromosome which represents a potential schedule.

The GA requires a few additional operators to function. These operators are methods necessary when working with encoded information:

3.3. Genetic Operators

3.3.1. Crossover

The GA uses crossover, where mating chromosomes are cut. Crossover is the most delicate operation of GA because it may produce unwanted irregular activity sequences within a job. A corrected 2-point crossover to avoid non-regular activity sequences of orders was used, which GOLDBERG (1989) refers to as a PMX-crossover operator (e.g. STEININGER 2007, p. 146 f.).

3.3.2. Mutation

Mutation describes the process of randomly changing values of a gene resulting in a variant form. This occurs with small probability. It is therefore 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 permutation.

3.3.3. Fitness

The fitness function is used to evaluate the fitness of each individual in the population and depends on the specific application. Generally a GA proceeds towards creating healthier individuals. If the fitness value is the only information available to the GA, the performance of the algorithm will be highly sensitive to the fitness function. Therefore when creating streamlined routines, fitness is the value of the objective function to be optimized.

3.3.4. Selection

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 method. An imaginary roulette wheel is constructed with a segment for each individual in the population. An individual's section size is based on their particular fitness value, with a fitter individual occupying a larger slice of the roulette wheel than a weaker one. 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. In all fitter individuals will have a propensity to be selected more frequently than weaker ones.

3.4. Genetic Algorithm Working Set Parameters

The GA needs a few additional parameters to work. These parameters specify the size of the population, the use of operators and so fourth.

3.4.1. Population size

The population size depends on the nature of the problem. Typically, it contains several hundreds or thousands of possible solutions. The population is generated randomly, making it possible to cover the entire range of possible solutions. Here a population size of 500 individuals will represent 500 feasible schedules.

3.4.2. Probability of crossover

The parameter probability of crossover affects the rate at which the crossover operator is applied. A higher crossover rate introduces new chromosomes more quickly into the population. If the crossover rate is too high, good individuals are eliminated faster than selection can produce improvements. A low crossover rate may cause stagnation due to the lower exploration rate.

3.4.3. Probability of mutation

Probability of mutation describes the likelihood that every gene of each individual in the new population will undergo a random change after a selection process. A low mutation rate helps to prevent any gene positions from getting stuck to single values, whereas a high mutation rate results in essentially a random search.

3.5. Final result

It is a characteristic of the GA that once fairly good solutions have been found their features will be carried forward into even better results, which will ultimately lead to a near-optimal solution. Therefore, GAs are particularly attractive for scheduling.

Compared with other optimization methods, GAs are suitable for traversing large search spaces since they can do this relatively rapidly and because the mutation operator diverts the method away from low scale optima. Being suitable for large search spaces is a useful advantage when dealing with schedules of increasing size since the solution space will grow very rapidly. It is important that this large search space is scanned as fast as possible to enable the practical and useful implementation of schedule optimization.

4. SIMULATION STUDY AND COMPARISON OF RESULTS

The simulation study was implemented according to the organization-oriented simulation method *FEMOS* (German acronym for production and assembly simulator) developed at the Institute of Human and Industrial Engineering of the University of Karlsruhe (Germany). *FEMOS* was used to analyze the entire production-logistical process chain (cf. ZÜLCH, GREINKE 2004; ZÜLCH, WARRISCH 2004, ZÜLCH 2008).

4.1. Test structure

The objective was to detect and define the relationships arising between the logistical sequence strategies and chosen logistical target parameters. In this analysis the results were compared to those of *REIMOS*. The test structure was based on empirical values which have proved viable during previous practical projects. The influencing factors, which were to change in the course of the simulation study, included the planned starting date (point of entry into the system), the planned finish date, the sequence of initialization of production orders or orders of a product group, respectively, plus different priority rules. The priority rules comprised a further selection of rules such as: first come first served, shortest operation time, longest operation time, slack time, and the earliest due date.

Furthermore, eight policies based on varying order initialization were examined. Policy S1 described the sequence of order initialization as specified by the industrial partner. S2 choose initialization according to product groups with increasing work content (the later the point of inclusion, led to the higher the work content). Here individual goods from the product group were combined equidistantly. S3 utilized the incorporation of individual products in equidistant intervals and the product groups were mixed. For S4 distinct incorporations in equidistant intervals occurred. These were arranged according to decreasing product groups with declining work content (the later the point of incorporation, led to the lower the work content). Individual products from the group were incorporated equidistantly. S5 described product groups with increasing work content (the later the point of incorporation led to the higher the work content). They were added one after the other whilst the individual products of the group are were combined at the same point in time. In S6 the order initialization of product groups occurred in equal intervals and the product groups were mixed. S7 looked at product groups with decreasing work content (the later the point of incorporation, led to the lower the work content). They were combined one after the other, with the individual products of a group incorporated at the same point in time. Finally S8 described block initialization, where all products were incorporated 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 capacity 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 meant that the operations rated with high priority were those at the end of the production process which would – according to the hypothesis – prove most effective with regards to the reduction of makespan and an increase in reliability of delivery.

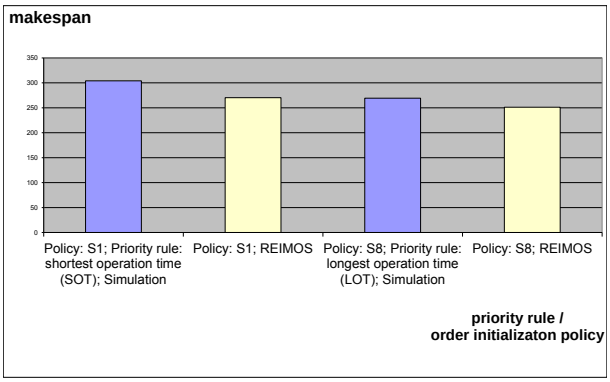

Figure 4. Comparison of makespans identified by means of *FEMOS* using order initialization policies and priority rules and those generated with the *REIMOS* planning method.

4.2. Results and comparison of results with *REIMOS*

The simulation runs were evaluated with *REIMOS* based on the target parameters makespan, delivery reliability and resource utilization. The results were then compared to those of the *REIMOS* planning method. The benchmark parameters employed were the lowest makespan of *REIMOS* and the simulation study achieved with order initialization policies 1 and 8. 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 8 allowed for the highest possible degree of flexibility as all orders are initialized at point t=0. This ensured a solid basis for the comparison of results from both methods. Out of the two policies, those with the best result in combination with the priority rules were selected. In addition, the simulation methods were compared with regards to the respective makespan of the product groups (block makespan). Figure 4 sets the makespan of the sequences identified by means of the *REIMOS* planning method alongside those generated by the *FEMOS* method. The strategies used were the priority rule shortest operation time in combination with order initialization policy 1 and longest operating time in combination with order initialization policy 8 respectively.

The comparison shows that *REIMOS* provided better results for both order initialization policies. The makespan achieved with *REIMOS* is approx. 11% lower for order initialization policy 1 and almost 7% lower for policy 8 compared with sequence planning on the basis

of priority rules. This proves that the more complex Genetic Algorithm is superior to the priority rules which are less challenging in terms of application.

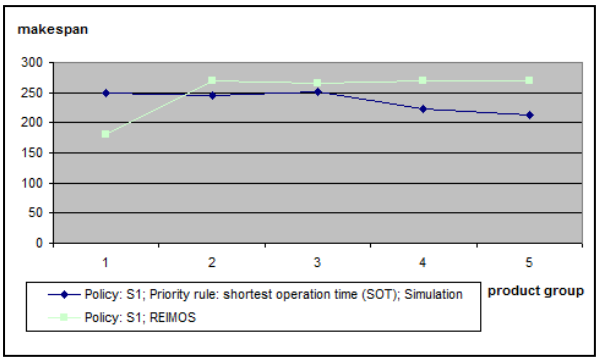

Figure 5. Comparison of makespans identified by means of *FEMOS* using order initialization policies and priority rules and those generated with the *REIMOS* planning method.

The diagram illustrated in Figure 5 is the product of a comparison of makespan for production orders of the 5 product groups generated with *REIMOS* and the simulated application of priority rules respectively. Within the simulation study, the comparison of makespan gave the best result for order initialization policy 1 in combination with the shortest operation rule. However order initialization policy 8 proved to be most effective when combined with the longest operation rule. Figure 4 shows a comparison of these simulation results with the results of *REIMOS*.

When the makespan of individual product groups for order initialization policy 1 was compared, it turned out that the values were 6% higher with *REIMOS* than when the priority rules were applied. The utilization of resources and the prioritization of orders were simulated. The best results of the simulation method were compared to *REIMOS*. In addition, the values were more evenly distributed with the exception of product group 1.

Comparing the makespan of individual production orders for order initialization policy 8 delivered the same results. The values of *REIMOS* were approx. 21% higher than those generated with the simulated use of priority rules. The individual makespan within one product group did not vary strongly from the group mean. This gives rise to the assumption that a correlation between an equidistant or almost equidistant spacing of interim arrival times within a product group and an improvement of makespan exists.

All of the above proves that the makespan results delivered by the use of priority rules fall short of those delivered by *REIMOS*.

5. CONCLUSION AND OUTLOOK

A computer algorithm based on the evolution of living beings may be surprising, but the extent to which this approach is applied is even more astonishing. Genetic algorithms have already proven their efficiency in many fields of application such as commercial, educational and scientific. Their usefulness in solving various kinds of problems have made them a preferable choice compared to other, more traditional approaches for a multi-criterion approach to target selection for modeling and evaluations.

The adaptation of a GA to schedule jobs in manufacturing workshops with time, resource and precedence constraints has been demonstrated here (STEININGER 2007). When using such GAs, it is often crucial to implement goal criteria. A new idea is to combine simulation and optimization processes. 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.

The simplicity of the methods used supports the assumption that GA can provide a highly flexible and user-friendly solution to the JSP. The use of standard software and an implemented "add-in" for Microsoft Project 2003 to realize the GA has shown that this approach can be profitable for solving real world scheduling problems (STEININGER 2007).

REFERENCES

Bagchi, T. P., 1999. *Multiobjective Scheduling by Genetic Algorithms*. Boston, Dordrecht, London: Kluwer Academic Publishers.

Bean, J. C., 1994. Genetic Algorithms and Random Keys for Sequencing and Optimizations. *ORSA Journal of Computing*. Hanover. 6 (2), 154-160.

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

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

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

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

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

Graham, R. L., Lawler, E. L., Lenstra, J. K. et al., 1979. Optimization and approximation in deterministic sequencing and scheduling. *Annals of Discrete Mathematics*. Amsterdam, 16 (5), 287-326.

Steininger, P., 2007. *Eine Methode zur Reihenfolgeplanung bei Mehrprodukt-Fertigungssystemen*. Dissertation, Universität Karlsruhe. Aachen (D): Shaker Verlag.

Taillard, É. D., 2006. *Scheduling instances*. http://mistic.heig-vd.ch/taillard/problemes.dir/ordonnancement.dir/ordonnancement.html. Status: 11.04.2011

Zülch, G.; Greinke, J. 2004. *Simulation-aided Reconfiguration of an Industrial Service System for the Repair of Electrical Tools*. Espoo: Sim-Serv.

Zülch, G.; Warrisch, W. 2004. *Simulation-aided Segmentation of a Mechanical Parts Manufacturing*. Espoo: Sim-Serv.