The International Journal of
TRANSPORT & *LOGISTICS*
Medzinárodný časopis
DOPRAVA A LOGISTIKA

ISSN 1451-107X

SOLVING JOB SHOP SCHEDULING WITH THE COMPUTER SIMULATION

Tomas Kloud [1], ***Frantisek Koblasa***[2]

[1] *Faculty of Mechanical Engineering, Department of Manufacturing Systems, TU Liberec, Studentska 2, 461 17 Liberec 1, Czech republic, e-mail: tomas.kloud@tul.cz*

[2] *Faculty of Mechanical Engineering, Department of Manufacturing Systems, TU Liberec, Studentska 2, 461 17 Liberec 1, Czech republic, e-mail: frantisek.koblasa@tul.cz*

Abstract: *Computer simulation of discrete events is one of most used tools in the field of manufacturing design and logistics process design, but it is still underestimated in the field of the production scheduling. This article focuses on solving a job shop scheduling problem (JSSP) with computer simulation. There is briefly introduced possible scheduling and optimization by the genetic algorithm in the Plant simulation software. Algorithm is tested on theoretical examples of classical JSSP. Comparison is made to known problems optima. Scheduling software is also used to compare not only ability to reach result near the optimal one, but also the time span of the optimization.*

Key words: *Job Shop Scheduling Problem, simulation, Plant Simulation*

1 INTRODUCTION

The importance of the production scheduling optimization is growing due fact that most of the companies developed its production management style based on more or less suitable solutions as the Material Requirements Planning (MRP), the Manufacturing Resource planning (MRPII), the KANBAN and the Theory Of Constraints (TOC) etc., so there is not much room to optimize the production by this way.

The Advanced Planning and Scheduling systems (APS) are using the constrained scheduling and the optimization. They are user friendly in the view of daily use on one side, but on the other side their modifications are very expensive.

The computer simulation is more demanding on the user skill and yet most of the simulation software are not very useful for the production scheduling. Their ability to describe production constraint accurately predetermines them to schedule and optimize hard production models as Job Shop Scheduling Problem (JSSP) [1, 2].

That is why we tried to implement the Active schedule generation and test it on the theoretical example in the simulation software.

The second chapter describes way to improve scheduling in the simulation software using the Active schedule generation. There are shown basic differences between the Active and Non-delay schedule generation and the basic between the simulation and the scheduling which must be considered implementing the Active schedules.

The third chapter shows possible way to implement the Active schedule in the Plant simulation software (Siemens PLM).

The fourth chapter is dedicated to the priority rules and Genetic Algorithm (GA) optimization. There is shown the comparison between results of Non-delay and Active schedules on the JSSP theoretical problems.

2 ACTIVE AND NON-DELAY SCHEDULE GENERATIONS FOR JSSP

The Job Shop Scheduling is one of the most popular and generalized production systems, which are hard to solve thanks to their non-polynomial hard nature. The classical JSSP problem can be described as follows [3].

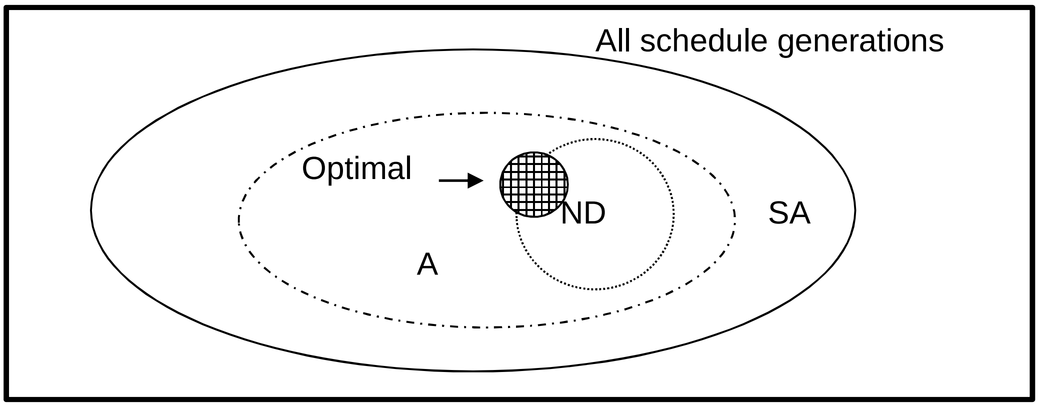
There are a set of M machines and a set of N jobs. Each job consists of a sequence of operations, each of which needs to be processed during an uninterrupted time period of a given length on a given machine. Each machine can process at most one operation at a time. It is assumed that any successive operations of the same job are processed on different machines.

There are three major kinds of the feasible schedule generation in the field of schedule optimization that we use for JSSP [4]:

- Semi-Active
- Active
- Non-Delay

The Semi-Active schedule is the schedule, where it is not possible to schedule the operation earlier without changing the sequence in which they are entering the machine.

The Active schedule is than the schedule, where is not possible to create the schedule by changing the order of the operation by starting the operation earlier without delaying other one [5]. This schedule generation is the most used in the optimization because the optimal schedule is always the Active one same as the Semi-Active and in the same time it is the subset of the Semi-Active schedules. So it gives us much smaller searching neighborhood to search than the Semi-Active ones. The last mentioned schedule generation is Non-Delay, which is the subset of the Active schedules (see Fig. 1). In this schedule no machine is idle (without assigned job), when the operation is available.



SA – Semi-Active schedules; A - Active schedules; ND – Non-Delay schedules

Fig.1 *Schedule generation map*

2.1 Active and Non-delay schedule generation algorithm

The algorithms for Active and Non-delay schedules could be described by following pseudo code [6] going through t stages:

P_t - the partial schedule of the *(t-1)* scheduled operations;

S_t - the set of operations schedulable at stage *t*, all the operations that must precede those in S_t are in P_t;

σ_k – the earliest time that operation O_k in S_t could be started

ϕ_k - the earliest time that operation O_k in S_t could be finished (1), that is

$$\phi_k = \sigma_k + p_k \tag{1}$$

where p_k is processing time of the O_k.

Step 1 – Let $t = 1$ with P_1 being null. S_1 will be the set of all operations with no predecessors

Step 2 – Find for Active (2)

$$\phi^* = \min O_k \text{ in } S_k\{\phi_k\} \tag{2}$$

or for Non - Delay (3) schedule

$$\sigma^* = \min O_k \text{ in } S_k\{\sigma_k\} \tag{3}$$

and the machine M^* on which ϕ^* (Active) or σ^* (Non-delay) occurs, if there is choice of M^* choose randomly.

Step 3 Choose an operations O_j in S_k which can be processed on M^* and satisfy (4) for Active

$$\sigma_j < \phi^* \tag{4}$$

or (5) for Non-Delay schedules

$$\sigma_j = \sigma^* \tag{5}$$

Step 4 Move to next stage by adding O_j to P_t and creating P_t+1; deleting O_j from S_t and creating S_t+1 by adding the operation that directly follows O_j in its job (unless job is finished); increment *t* by 1

Step 5 If There is operation left unscheduled that go to step 1 else Stop

Seeing these algorithms, is obvious, that main difference is in the Step 2 resp. Step 3, where Active schedule is searching for further ending time and Non delay schedule for starting time.

In spite of that Active schedule generation can always find optimal solution, is not used frequently in the real world practice due two major reasons:

- The earliest finishing time search – we are not able to predetermine exactly the finish time of an operation due to real world delays (Machine breaks, worker failures, transport delays etc.)
- It is hard to explain fact that, by delaying operation, which is physically available to process and by giving priority of an operation, that is still processed on another machine, we can shorten the overall schedule time (makespan)

Non-Delay schedule generation is often used in spite of their lack of ability to get to the optimum in the case of real world problems and in the simulation software.

2.2 Scheduling and simulation

The basic difference (see Fig. 2), which has to be considered, is that simulation realized time for jobs and machines simultaneously and scheduling separately. So simulation considers scheduling opportunity as Non-Delay schedules – operation is available for processing when both job and machine are physically available too. Scheduling considers operation available when preceding operation was scheduled (satisfying machine and job constraint).

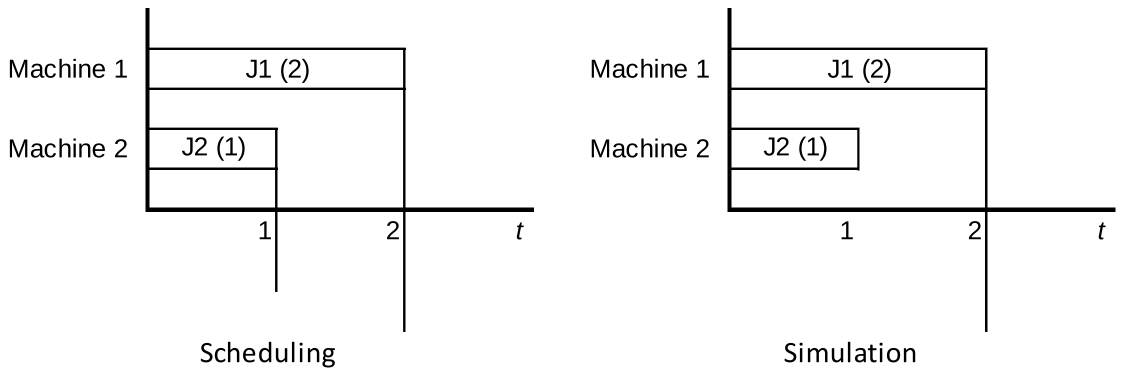


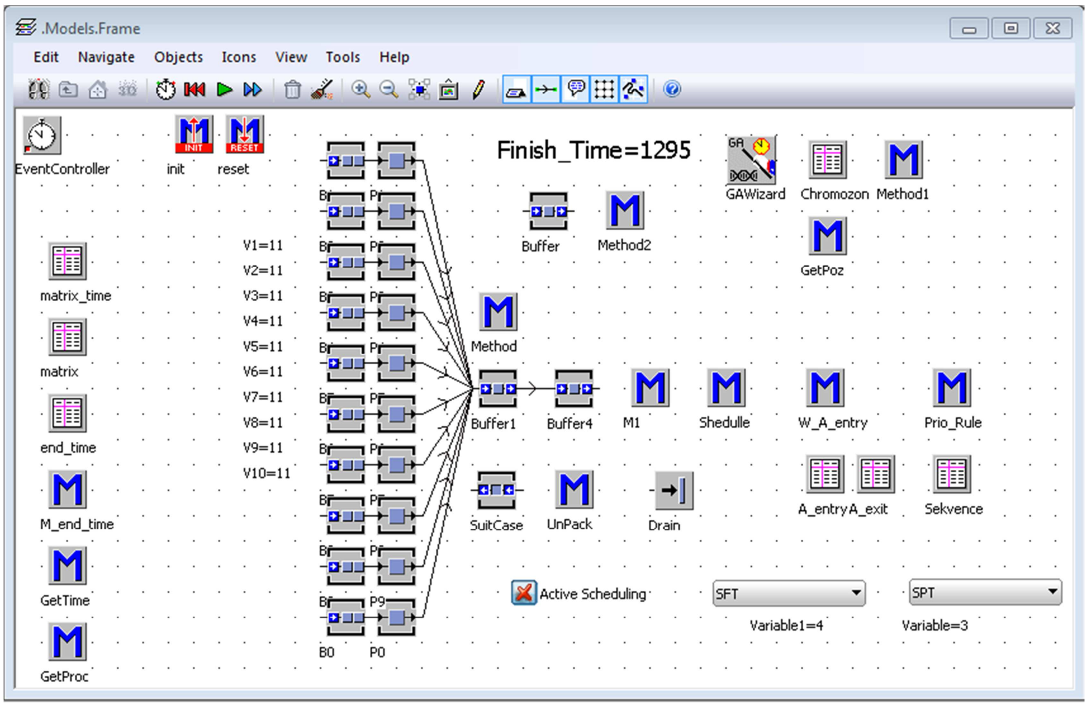
Fig. 2 *Scheduling a simulation time comparison*

There is *t = 2* for Machine 1 resp. Job 1 and *t = 1* for Machine 2 resp. Job 2, in the case of scheduling. The simulation has *t = 2* equal for whole system.

The main thing implementing active schedule generation in to the simulation is than keeping both Machine and Job time considering the system time.

3 THE IMPLEMENTATION OF THE ACTIVE SCHEDULE GENERATION

Plant Simulation is the simulation tool made by Siemens PLM, which uses object oriented modeling. It is used only basic elements (see Fig. 3) of the Plant simulation V10 to make theoretical models and algorithms described in the second chapter. There are used priority rules and optimization wizard for genetic algorithm to optimize schedules, which is implemented in higher versions of the Plant simulation.



Finish_Time=1295

V1=11
V2=11
V3=11
V4=11
V5=11
V6=11
V7=11
V8=11
V9=11
V10=11

SFT
Variable1=4

SPT
Variable=3

***Fig. 3** Illustration model of FT10*

3.1 Model Elements

The whole Job shop scheduling problem together with Active and Non delay algorithm is modelled by these elements:

SingleProc
- P0 – P9 (Machine representation)

PlaceBuffer
- B0 – B9 (Warehouse in front of machines)

Checkbox
- Checkbox – Active Schedule (Use to select Active schedule generation)

DropDownList
- DropDownList, DropDownList1 (Selection of the scheduling rule)

Drain
- Drain (Element exit)

Buffer
- Buffer (Generation of the jobs and initiation of Method 2 on exit)
- Buffer1 (Initiate method Unpack on entrance and Method on exit)
- Buffer4 (Initiate method M1)
- SuitCase (Buffer used for waiting jobs)

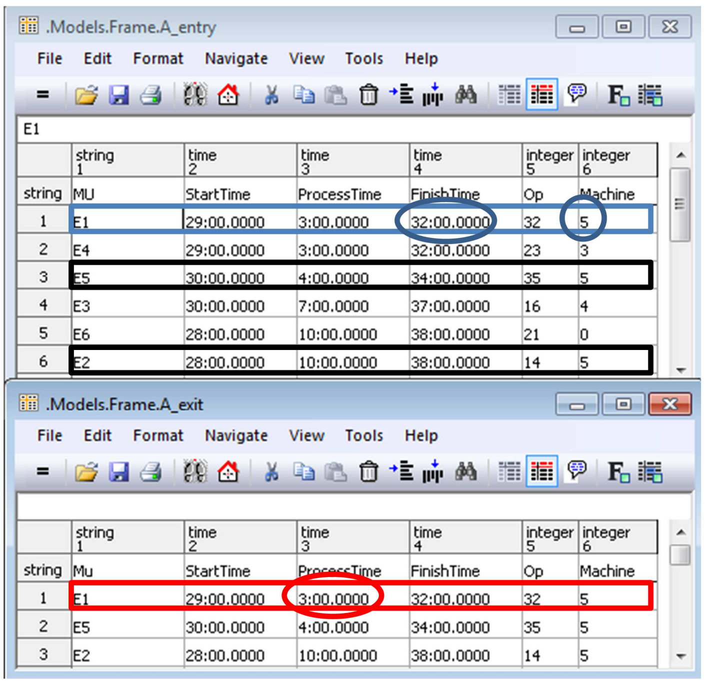
TableFile
- Matrix (Technology order – machine routing)
- Matrix_Time (Processing times of the operations)
- End_Time (Holds time availability of the machines and jobs)
- A_Entry (Available operation evidence)
- A_Exit (Operations selected by scheduling rule evidence)
- Sekvence (Sequence generated by scheduling rule)
- Chromozon (Holding sequence of all operations – used by genetic algorithm)

Variable
- V1 – V10 (job progress identification)

- Finish_Time (Makespan, Fitness function value)
- Variable (Chosen by selection in the DropDownList, identify which column from the A_Exit will be used for sorting)
- Variable1 (Same as Variable 1 but for A_Entrance)

Method

- Init (Starts at the begining of the simulation, generates jobs to the Buffer)
- Reset (Resets model tables except Matrix and and Matrix_time)
- M_end_time (Writes ending time (table end_time) for job and occupation time for machine)
- GetTime (Returns values from the table Matrix_time)
- GetProc (Returns values from the table Matrix)
- Method (Checks number of the operation, if is operation number higher that overall number of operations than moves entity to the drain else if following Buffer 5 is busy than moves entity to the buffer “SuitCase”)
- Method1 (Generates Chromosome table)
- Method2 (Writes attributes to the entities (number of operation and sequence of the technological order) and writes process information to table A_Entry (start time, process time, end time, machine,)
- UnPack (checks fulfilment of the buffer Suitcase, if there are any entities, than they are moved back to buffer1)
- M1 (method assigns entities to the appropriate sources (machines))
- Schedule (Initiates scheduling algorithm. Initiates repeatedly W_A_entry and Prio_Rule, which are selecting operations and records them to the Sequence)
- W_A_Entry (Sorts A_entry table by the variable (Variable1), selects conflict machine)
- Prio_Rule (Creates the conflict set of the operations, selects operation by variable1, writes result in to the Sequence table, resets E_exit table, initiates method M_end_time and adapts table A_entry – Start time, Finish time of affected operations (see Figure 4)
- GetPoz (Gets row number from Chromosome table by the operation sequence)


.Models.Frame.A_entry
File Edit Format Navigate View Tools Help
.Models.Frame.A_exit
File Edit Format Navigate View Tools Help

E1

	string 1	time 2	time 3	time 4	integer 5	integer 6
string	MU	StartTime	ProcessTime	FinishTime	Op	Machine
1	E1	29:00.0000	3:00.0000	32:00.0000	32	5
2	E4	29:00.0000	3:00.0000	32:00.0000	23	3
3	E5	30:00.0000	4:00.0000	34:00.0000	35	5
4	E3	30:00.0000	7:00.0000	37:00.0000	16	4
5	E6	28:00.0000	10:00.0000	38:00.0000	21	0
6	E2	28:00.0000	10:00.0000	38:00.0000	14	5

	string 1	time 2	time 3	time 4	integer 5	integer 6
string	Mu	StartTime	ProcessTime	FinishTime	Op	Machine
1	E1	29:00.0000	3:00.0000	32:00.0000	32	5
2	E5	30:00.0000	4:00.0000	34:00.0000	35	5
3	E2	28:00.0000	10:00.0000	38:00.0000	14	5

Fig. 4 *Sample of the operation selection*

3.2 Scheduling logic

The previous described model than uses this scheduling logics:

1) Generating jobs by the Matrix table – model initiation
2) Moving Job from the "Buffer" to the "Buffer1" and writing number of the operation and operation sequence by the Method2 to the A_Entry
3) Repeat step 2) until Buffer is not empty
4) Schedules during the first run through Buffer4 :
   a. Writes the first operations to the A_Entry table
   b. Select operation
   c. Makes conflict set from the operations in the A_Exit table
   d. Schedules operation by Priority rule or GA
   e. Records sequence
   f. Adapts end time by process time
   g. Adapts both job and machine availability respecting end time
   h. Back to the step 2) until there are any operations in the A_entry table
   i. Search for highest finish time (Table Sequence) and writing value in to the Finish_Time (finding makespan), simulation ends

3.3 Schedule verification

There were used priority rules to verify and validate both models and scheduling methods at first, because makespan of the tested models where already known [2]. Considering that same model with same priority rule and schedule generation should give us same result (makespan and schedule). There were tested all theoretical models and same results where achieved only in two models (FT6 and FT 10). There were tested gained results systematically thanks to provided Gantt chart and it was found that priority rule Shorter Processing time is not suitable to validate and verify models. The reason is that even after using priority rule two or more operations remains and random selection is than applied (See Figure 5). There are 2 jobs. First Job goes at first at machine 1 with processing time of 8 and then to the machine 2 with processing time 16. Second Job goes at first at machine 1 with processing time of 8 and then to the machine 3 with processing time 4

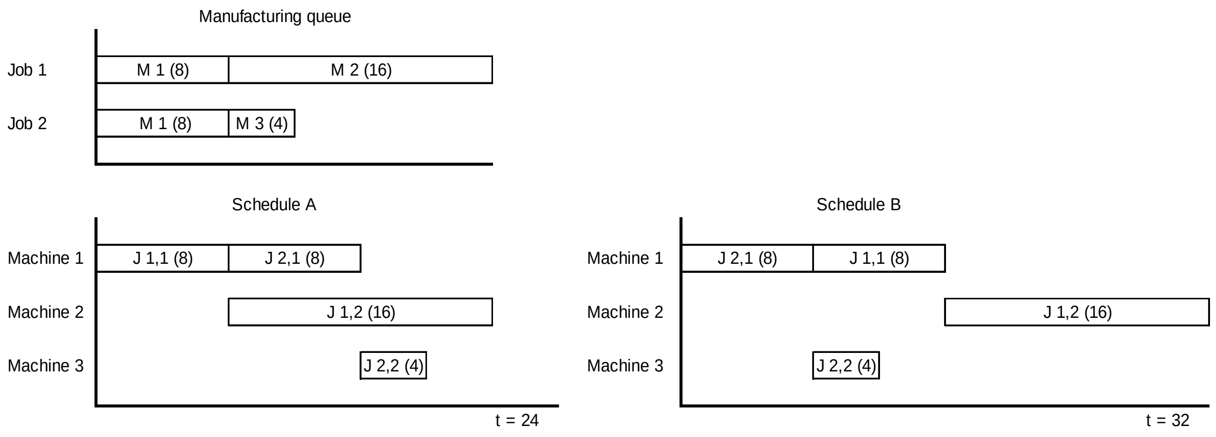


***Fig. 5** Different schedules with same priority rule*

Two operations with all same starting, finishing and processing time occur, so random selection is applied. Both makespan and Gantt chart are different using same priority rule.

4 SCHEDULE OPTIMIZATION BY THE PRIORITY RULES AND GA

Objective function for optimization experiments is makespan, which is the time to complete all jobs. Priority rules and Genetic Algorithm using Active and Non-delay schedules where tested on 10 well known instances of Job Shop Problem (the first number indicates number of jobs and the second indicates the number of machines):

- H. Fisher, G.L. Thompson [7]: 6x6 (ft6); 10x10 (ft10); 20x5 (ft20).
- R.H. Storer, S.D. Wu, R. Vaccari [8]:50x10 (sw11).
- S. Lawrence [9]: 10x5 (La2); 10x10 (La19); 15x10 (La21); 20x10 (La27); 20x10 (La30); 15x15 (La40).

There were used two priority rules Shorter processing time (SPT) and First In First Out (FIFO), which are most common in the industrial practice in the terms of MRPII based systems or by foreman.

Genetic algorithm setting parameters are set as:

- Population = 2x Total number of operations +100
- Cross over: Order crossover (OX)
- Cross over coefficient : 0.8
- Mutation: Random up to 0.4
- Challenge rule: Offspring 1 of 4
- Number of generations: 100

Table 1 shows comparison of the results given by Priority rules (First In First Out, Shorter processing time and genetic algorithm for each scheduling generation.

***Tab. 1** Makespan comparison (in time unites)*

Theoretical problem	**Optimum**	**Non-Delay**			**Active**		
		SPT	FIFO	GA	SPT	FIFO	GA
FT6	55	88	75	57	94	75	55
FT10	930	1171	1295	974	1429	1295	1051
FT20	1165	1374	1614	1198	1728	1614	1332
La02	655	802	812	668	1019	812	711
La19	842	959	1001	876	1413	1001	938
La21	1046	1284	1792	1098	1792	1266	1263
La27	1235	1798	1629	1350	2309	1629	1471
La30	1355	1721	1513	1362	2165	1513	1468
La40	1222	1488	1441	1289	1789	1441	1347
sw11	2983	4063	4549	3330	4549	4399	3355

Results given by GA are much better in the terms of objective function makespan as it was expected. The cost of the optimization by GA is much higher (see Table 2) than by the Priority rules (time format is Days-hours-minutes-seconds).

Tab. 2 *Timespan of the optimization by GA*

Theoretical problem	**Timespan of the optimization**	
	ND	A
FT6	0:23:41	0:28:08
FT10	1:41:08	1:14:04
FT20	1:57:46	1:19:57
La02	0:45:36	0:35:51
La19	2:05:18	1:56:30
La21	3:02:15	2:50:16
La27	5:48:27	6:01:04
La30	13:39:39	13:36:03
La40	14:37:26	14:51:31
sw11	2:16:47:06	2:15:05:25

Solving schedule by FIFO and SPT where obtained in seconds, GA takes from minutes to days. This high timespans are also caused by the ending conditions of the GA which is as mentioned before - number of generations. Nearly the same results in the much shorter timespan could be obtained by other ending conditions as the number of the generations without the improvement. The best result for LA21 in the case of the Active schedule is met during 91st generation, but nearly the same result (1278) in the 54th (see Fig. 6). The timespan of the Non-Delay solution could be much shorter reaching best solution 46th generation (see Fig. 7).

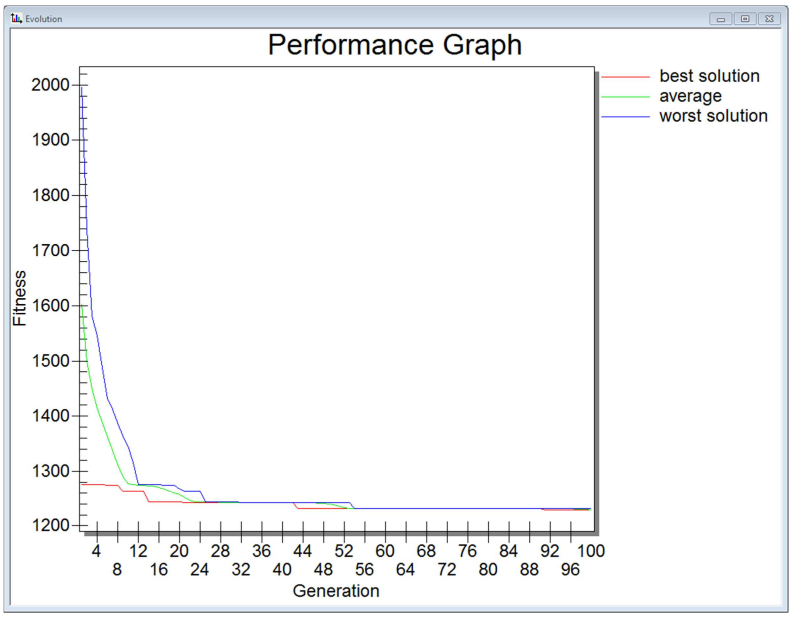


Fig. 6 *Performance graph of the LA21 Active schedule generation*

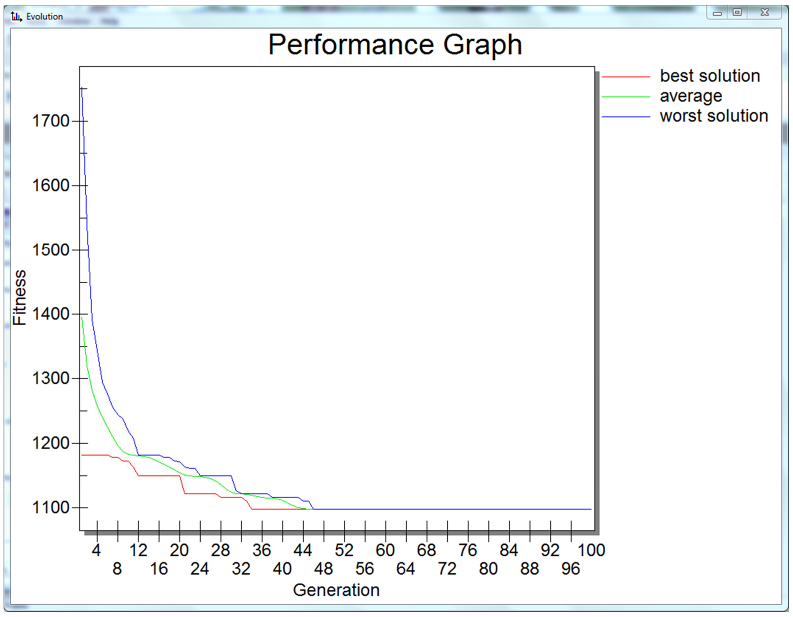


Fig. 7 *Performance graph of the LA21 Non-Delay schedule generation*

We tried to improve optimization in the Plant simulation by implementing Active schedule generation expecting, that it will improve objective function thanks to its ability to reach optimal solution. Table 3 shoves deviation from the optimum and average overall deviation. Thus Active the schedule reached optimum in the FT06 model, in the average, Non-Delay schedules reached better solution in the comparable time (see Tab. 3).

Tab. 3 *Deviation from the optimum*

Theoretical problem	**Deviation from the best solution**	
	ND	A
FT6	3,51%	0,00%
FT10	4,52%	11,51%
FT20	2,75%	12,54%
La02	1,95%	7,88%
La19	3,88%	10,23%
La21	4,74%	17,18%
La27	8,52%	16,04%
La30	0,51%	7,70%
La40	5,20%	9,28%
sw11	10,42%	11,09%
	Average	
	4,60%	10,35%

5 CONCLUSION

The Active schedule generation could reach better solutions as expected but additional research on implementation in the simulation is required. The main problem is that searching space in the Active schedule generation is much greater than in the case of Non-Delay schedules. This searching space could be reduced by other methods as Critical Path Method, which is next step in our research.

Following research will be focused also on the ending conditions of the optimization and possible modifications of Genetic Algorithm in the Plant simulation software.

References

[1] MANLIG, F. Optimierung von Fertigungsprozessen mit Rechnersimulation In: Forschungsergebnisberichte 2001 der Professur für Produktionsautomatisierung Steuerungstechnik. [Forschungsbericht] TU Dresden, PAS, 2001.
[2] Koblasa, F. Dias, L.S. Oliveira, J.A. Pereira, G. Heuristic Approach as a way to Improve Scheduling in ERP/APS Systems. Proceedings of 15th European Concurrent Engineering Conference (ECEC2008). Eds. A. Brito and J.M. Teixeira, 47-51, Porto April 2008. EUROSIS-ETI Publication. ISBN 978-9077381-399-7. (ISI-Thomson and INSPEC referenced)
[3] Vaessens, E. Aarts and Lenstra, R. Job Shop Scheduling by Local Search," INFORMS Journal on Computing, vol. 8, pp. 302-317, 1996.
[4] Baker, K.R Introduction to Sequencing and Scheduling. John Wiley & Sons Inc. ISBN-13: 978-0471045557 (November 1974),
[5] Pinedo, M. Scheduling - Theory, algorithms and systems. Prentice Hall,Inc 1995. ISBN 0-13-706757-7
[6] Giffler, B. Thompson, G. Algorithms for Solving Production Scheduling Problems. European Journal of Operational Research, 1960, vol. 8, pp. 487-503
[7] Fisher, H. and Thompson, G.L. Probabilistic learning combinations of local job-shop scheduling rules", J.F. Muth, G.L. Thompson (eds.), Industrial Scheduling, Prentice Hall, Englewood Cliffs, New Jersey, 225-251. 1963.
[8] Storer, R.H. Wu, S.D. and Vaccari, R. New search spaces for sequencing instances with application to job shop scheduling. Management Science 38, 1495-1509. 1992.
[9] Lawrence, S. Resource constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement), Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pennsylvania. 1984.

Recenzia/Review: *doc. Ing. Gabriel Fedorko, PhD.*