UNIVERSITY OF LEIPZIG

Institute of Production Management
and Industrial Information Management

Marschnerstr. 31, 04109 Leipzig, Germany
Phone: [49] / (0)341 / 4941-182, Fax: -125

Report No. 17

The Threshold Waving Algorithm for Job Shop Scheduling

Jukka Siedentopf

⟨siedentopf@wifa.uni-leipzig.de⟩

- to appear -

August 1995

Index

Summary

A unidirectional mutation-selection approach for the general job shop scheduling problem is presented. The underlying algorithm uses threshold accepting as an iterative search technique and incorporates a special non-monotonic threshold function (*threshold waving*). Search is performed in the space of operation sequences and, for the purpose of evaluation, is combined with a simple heuristic transforming operation sequences into schedules. Mutation of solution candidates is based on a critical path analysis and, additionally, on a probabilistically employed local hill-climbing operator. Effectiveness of the approach is demonstrated within a comparison with some well-known heuristic and exact algorithms.

Remark: A slightly shortened and changed version of of this report has been submitted to the *Symposium über Operations Research*, the 1995 Annual Conference of the DGOR, the GMÖR, and the ÖGOR.

1 Introduction

In the field of combinatorial optimization an increasing number of approaches introducing some kind of stochastic search are recommended as alternatives to somewhat more conventional approaches of mathematical programming. A common characteristic of many of these approaches is the principle of generating new solution candidates (configurations) by performing some (often stochastic) perturbations (mutations or modifications) of already available ones. Another characteristic is that the selection of configurations permitted to remain in the solution process does not only depend on the quality of the solution represented by the considered configuration, but also on external parameters.

The number of configurations involved in the solution process can serve as a separation criterion for the proposed approaches: In *multidirectional* approaches like genetic algorithms (GA) [e.g. 11] a multitude of configurations is managed in a population. Availability of several different configurations is an indispensable requirement for the recombination of new configurations from the building blocks of already available ones. Recombination is the predominating principle for generating configurations in genetic algorithms. Mutation serves for adding new building blocks by implementing small stochastic perturbations in the available building material.

In another class of more simply structured *unidirectional* approaches, e.g. simulated annealing [e.g. 1] or derivates like threshold accepting [6, 7], only one configuration undergoes a process of mutation. Thus, selection is reduced to a decision whether a new, modified configuration will replace the old one.

Facing complex scheduling problems with approaches using recombination it is a major problem to find a suitable representation of schedules: By means of reducing the search space, the desired structure should be capable of representing schedules effectively, i.e. maximizing the ratio of the number of feasible schedules to the number of representable ones. Simultaneously, in order to exploit the implicit parallelism of multidirectional approaches, the availability of a suitable recombination operator working on the proposed structure is presupposed. Shortcomings of the ‘team-work’ of a representation structure and the corresponding recombination operator frequently causes inefficiencies: A lot of time is spent either searching the space of infeasible solutions, e.g. in case of binary representation, or recombining solutions without gaining solution quality. As long as ‘efficient’ combinations of representation structures and recombination operators are missed, unidirectional approaches seem to be more fruitful for the development of fast algorithms for scheduling purposes - despite some discouraging experiences made with the ‘logarithmically cooled’ simulated annealing. In the following, such a unidirectional approach for job shop scheduling considering the objective of makespan minimization is introduced.

2 Threshold waving

2.1 The algorithm

A first version of the sketched algorithm [13] has been developed by means of a comparison with a specific genetic algorithm [12]. The algorithm adopts components of the GA, avoiding the originally used binary representation structure which only serves for applicability of the standard recombination operator (crossover). The algorithm uses threshold accepting as a base search strategy very similar to simulated annealing, deviating in particular in the acceptance of new configurations: A modified configuration serves as a working basis for further search if its solution quality is better than or not more than an actual threshold worse than that one of the current configuration. Hence, in contrast to simulated annealing threshold accepting does without probabilities of acceptance and the formulation of a cooling or annealing schedule - a very critical task when applying simulated annealing [e.g. 1] - is replaced by a slightly more simple instruction of a threshold decrease. Additionally, a monotonic and linear decreasing threshold function is used and the threshold is decreased after a fixed number of trials (modifications).

In the resulting threshold accepting algorithm for makespan minimization (TAMM) solution candidates are coded as task or job sequences, entered e.g. in a matrix row by row for the individual machines. An initial solution is created by randomly generating such sequences. In order to generate schedules, the task sequences serve as input for a simple heuristic schedule builder together with the given machine sequences of the jobs and the given operation times. The schedule builder tries to build up a schedule according to the proposed sequences. In cases of mutually blocking sequences (i.e. it is not possible to schedule any operation on any machine), a repair algorithm changes one of the sequences giving priority to an operation currently to be processed on the selected machine while shifting as few other operations as possible.

Significant improvements of the TAMM algorithm have been attained by implementing two extensions: Firstly, the modification operator, generating neighbourhood solutions by exchanging two randomly chosen elements in the operation sequence of a randomly chosen machine, is restricted to successive operations lying on a critical path of the considered schedule. Secondly, an operator is added performing local hill-climbing by modifying the task sequences in a randomly chosen order. Within this operator only modifications resulting in higher solution qualities are accepted. Each sequence is modified as long as any improvement is attained. Because each modification causes a rebuilding of the schedule and, in case of improved quality, maybe even of the critical path, this local search operator is very expensive and therefore should be used carefully.

Some first tests of that improved version of TAMM concerned the shape of the threshold function to be used within a given runtime limit, i.e. the combination of a threshold's start value and the number of trials permitted at each threshold level. The range of possible combinations is limited by the given runtime which can be translated into an overall number of trials permitted.

As a result of the considered tests, combinations of high threshold's start values and low numbers of trials permitted at each threshold level clearly outperformes reverse combinations. Or, alternatively spoken, compared to the 'threshold schemes' originally used in TAMM, the number of trials needed to satisfy a given solution quality's level can be reduced using high threshold's start values and low number of trials permitted at each threshold level. Clearly, this result expresses only a tendency without claiming general validity. Nevertheless, it led to a third extension of the TAMM algorithm, considering the use of the 'released' number of trials in a way that the solution quality is further improved: The rule of a monotonic decreasing threshold is replaced by a threshold waving down and up between an (initial high) upper level and zero with a decreasing amplitude, i.e. an upper level decreasing from 'wave to wave'. Thus, the new algorithm is called threshold waving (TW). The TW algorithm for job shop scheduling is presented in Figure 1 (for the sake of simplicity it is supposed that the initial value of *threshold_upper_limit* is positive).

2.2 Parametrization

In detail, TW can be controlled by adjusting the following parameters (not all explicitly listed in Figure 1): The starting value (*thresholdstart*) of the threshold's upper limit (*threshold_upper_limit*) as well as the number of trials after which the threshold is reduced or raised (*normal_wavefactor*).

Sometimes it seems to be advantageous to stretch the search of regions of slow convergence, i.e. of high solution quality levels and low threshold's upper limits. Therefore, a parameter *max_wavefactor* is introduced replacing the normal wavefactor when the threshold's upper limit falls below a value *wavefactor_limit*.

Considering the modification operator, the number of random operation exchanges performed each time the operator is called is bounded by the value of *max_modification_factor*. An exact (integer) value is, also randomly, chosen from the interval [1, *max_modification_factor*].

Finally, due to its high cost, the application of the local search procedure is doubly controlled: Firstly, permanent local search (i.e. follwing each single modification) is permitted only after reaching a threshold's upper limit of *local_search_limit*. Secondly, a probability for the application of local search is given by the value *local_search_probability*, independent from the threshold or the threshold's upper limit.

```
THRESHOLD WAVING
initialize  candidate, threshold_upper_limit, threshold ← threshold_upper_limit,
            wavefactor, trials ← 0, temp_optimum ← candidate
compute makespan (candidate) within BUILD_SCHEDULE (candidate)
determine CRITICAL_PATH (candidate)
loop  new_candidate ← MUTATION (candidate)
      compute makespan (new_candidate) within BUILD_SCHEDULE (new_candidate)
      determine CRITICAL_PATH (new_candidate)
      randomly or deterministically perform LOCAL_SEARCH (new_candidate)
      trials ← trials + 1
      ΔE ← makespan (candidate) - makespan (new_candidate)
      if ΔE > 0 then temp_optimum ← new_candidate
      if ΔE > (-1)·threshold then candidate ← new_candidate
      if trials = wavefactor then
            trials ← 0
            if threshold = 0 then
                  threshold_upper_limit ← threshold_upper_limit - 1
                  if threshold_upper_limit = 0 then th_adder ← 0
                  else th_adder ← 1
            elsif threshold = threshold_upper_limit then th_adder ← (-1)
            threshold ← threshold + th_adder
until (threshold = 0) and (threshold_upper_limit = 0)
solution ← temp_optimum
```

Figure 1: Threshold waving algorithm

3 Computational results

Tests are performed by using two well-established data sets of Fisher and Thompson [9]: The processing of 10 jobs on 10 machines (‘10x10-problem’) as well as of 20 jobs on 5 machines (‘20x5-problem’). The optimum makespan values are 930 for 10x10-problem and 1165 for 20x5-problem, respectively. The parameter settings used for TW are listed below in Table 1. The last row indicates the number of repetitions (*runs*) of TW with the respective parameter setting.

The cited parameter were determined in a trial and error process trying to fit an arbitrarily fixed mean runtime value of 30 seconds for the 10x10-problem. The attained settings resulted in a mean runtime of about 45 seconds for the 20x5-problem and have then been adopted (again by trial and error) to lead to solution qualities as good as possible within the given time frame.

	10x10-problem	20x5-problem
thresholdstart	95	105
normal_wavefactor	1	3
max_wavefactor	5	3
wavefactor_limit	3	0
max_modification_factor	2	2
local_search_limit	3	0
local_search_probability	0.2	0.02
runs	1,000	1,000

Tab. 1: Parametrization

Computational results are presented in Table 2 and refer to an implementation of TW written in Pascal on a Pentium-based PC (90 MHz). The distributions of the solution qualities for both problems are shown in Figure 2 and Figure 3. The values of the worst solutions obtained are 1034 for the 10x10-problem and 1282 for the 20x5-problem, respectively.

	10x10-problem	20x5-problem
best found solution quality (makespan)	930	1165
mean solution quality (makespan)	973.93	1203.54
average deviation from optimum [%]	4.72	3.31
standard deviation	20.06	19.15
mean runtime$_{total}$ [sec]	30.14	45.18
mean runtime$_{best}$ [sec]	18.14	36.44
shortest time for detecting optimum [sec]	2.80	38.94
mean initial solution quality	3339.37	3589.13
correlation (initial and best solution quality)	≈ 0.0004	≈ - 0.0003

Tab. 2: Computational results

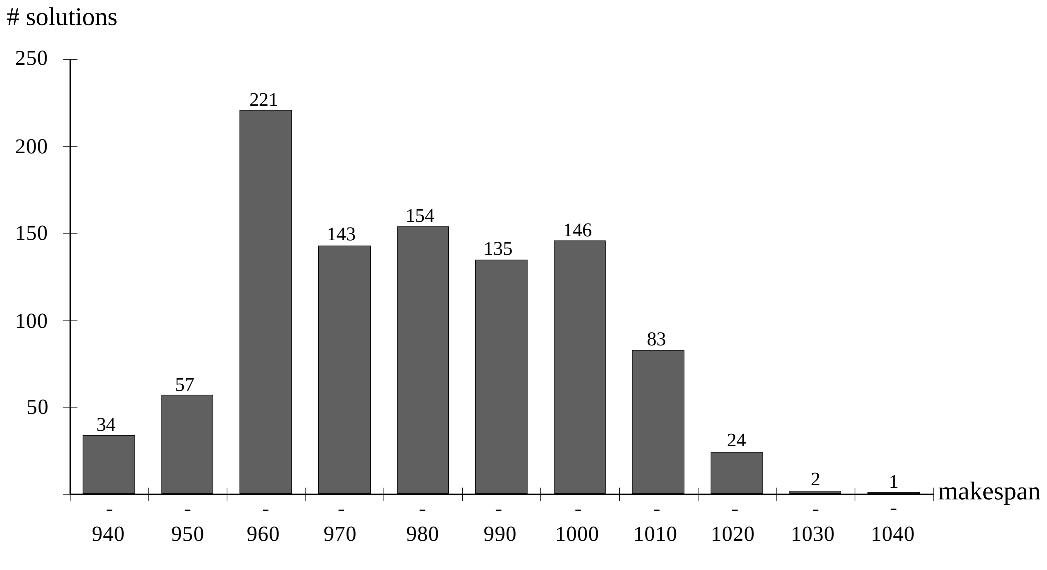

Fig. 2: Distribution of solution qualities for the 10x10-problem

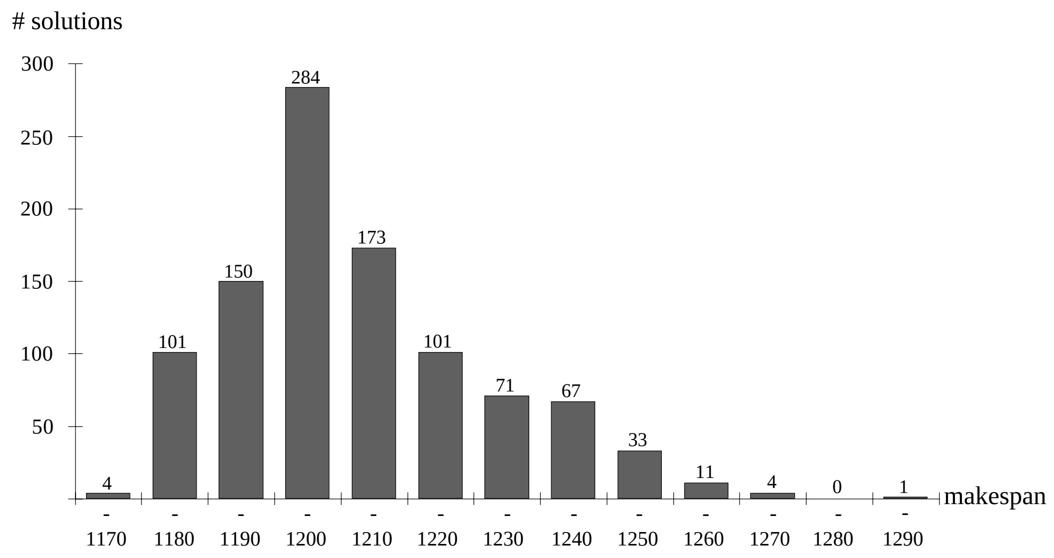

Fig. 3: Distribution of solution qualities for the 20x5-problem

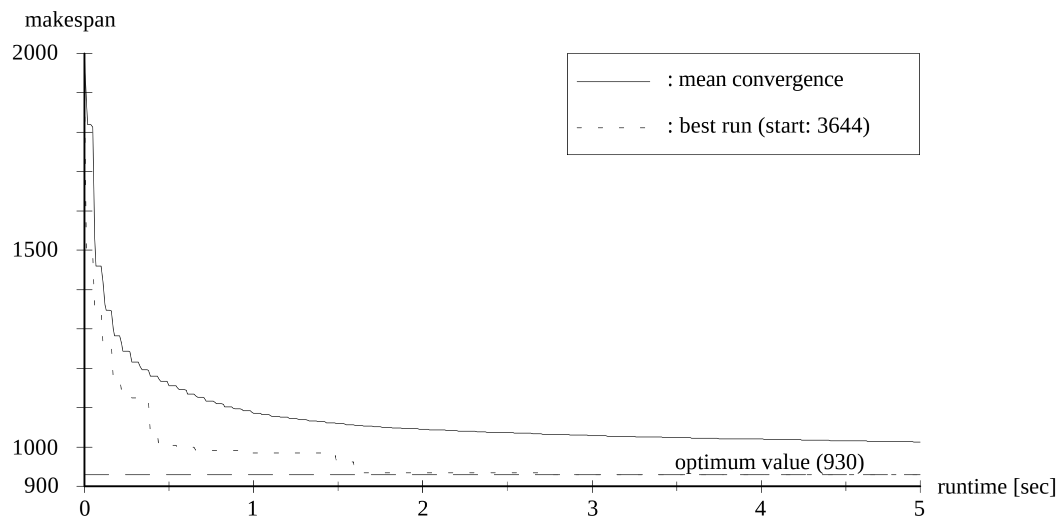

Fig. 4: Convergence of TW for the 10x10-problem

Remarks: (1) Beside the waving threshold, a very steep threshold function is the most significant characteristic of TW, appearing in only 1 to 3 trials per actual threshold using the cited parametrization. (2) Note the very short time of 2.8 seconds for detecting the optimum value of 930 for the ‘famous’ 10x10-problem. This value, even if supported by a stroke of luck, illustrates the strength of the convergence properties of TW (see Figure 4). (3) It should also be mentioned that the used schedule builder does not produce non-delayed or other active schedules (just semi-active ones). Nevertheless, comparisons to the algorithm of Giffler and Thompson [10], producing active schedules, revealed the superiority of the used algorithm, indicated by a significantly better convergence [13]. (4) A test using the parameter settings of the 20x5-problem for solving the 10x10-problem led to a mean solution quality of 966.81, a standard deviation of 14.88, and a mean runtime of 40.03 seconds. Use of parameter settings of the 10x10-problem for solving the 20x5-problem led to a mean solution quality of 1212.55, a standard deviation of 24.70, and a mean runtime of 43.98 seconds. (5) Neglecting local search, the runtime of TW is mainly determined by the number of trials working as a factor for the runtime(s) of the algorithm(s) for building up schedules and critical paths. Let *a* be the *normal_wavefactor*, *n* the *thresholdstart*, *b* the *max_wavefactor*, and *s* the *wavefactor_limit*. The number of trials is then given by: $[a \cdot (n^2+1) + (b-a) \cdot (s^2+s+1)]$.

4 Comparison

Concluding, the presented results are compared to the results of some other authors using the same ‘problem artefacts’ of Fisher and Thompson:

- the branch&bound algorithm of Barker and McMahon [3] (BM)
- the branch&bound algorithm of Carlier and Pinson [4] (CP)
- both shifting-bottleneck-procedures of Adams, Balas and Zawack [2] (SB1, SB2)
- two (hybrid) genetic algorithms of Dorndorf and Pesch [5] (DP1, DP2)
- the genetic algorithm of Fang, Ross and Corn [8] (FRC)
- the simulated annealing approach of Van Laarhoven, Aarts and Lenstra [14] (SA1,...,SA4)
- the TAMM algorithm [13] (TAMM)

Table 3 summarizes solutions qualities and runtimes obtained by the mentioned approaches, a selected parametrization of TAMM from [13] and TW. Runtimes values are rounded off, if necessary, and found optimum values as well as corresponding runtime values are marked by bold types. Because of the close relation to TW, all four parametrizations proposed in [14] are listed for the algorithm SA.

	10x10-problem		20x5-problem	
	best / mean solution	runtime [sec]	best / mean solution	runtime [sec]
BM	960	193	1303	132
CP	**930**	**3305**	**1165**	**1234**
SB1	1015	10	1290	3
SB2	**930**	**851**	1178	80
DP1	960	932	1249	1609
DP2	938	106	1178	95
FRC	949	< 1500	1189	< 1800
SA1	1028 / 1040	113	1325 / 1354	123
SA2	951 / 986	779	1184 / 1229	848
SA3	937 / 942	5945	1173 / 1187	6840
SA4	**930** / 933	**57772**	**1165** / 1173	**62759**
TAMM	**930** / 1007	**194**	**1165** / 1213	**265**
TW	**930** / 974	**30**	**1165** / 1203	**45**

Tab. 3: Comparison of solution qualities and runtimes

Remark: It should be noted that, due to different computer systems serving as bases for implementations and tests of the different approaches, no direct comparability of the stated runtime values is given. The following computer systems are indicated by the authors: BM: Cyber 171, CP: PRIME 2655, SBx: VAX 780/11, DPx: DECstation 3100, FRC: SUN-4, SAx: VAX 785, and TAMM: PC 80486 (66MHz).

5 Outlook

With regard to the practical applicability, the effectiveness of an approach is just a necessary, not, however, a sufficient condition. A proof of adaptability and flexibility of TW by applying it to different real-world problems should be the next step creating a versatile algorithm as a useful tool for the development of operational systems. Due to the multitude of introduced parameters, the parametrization of TW is a problem of its own, suffering from the combinatorial 'explosion' of possible parameter combinations. Thus, another requirement for further developments is the implementation of an automated parameter tuning, perhaps exploiting possibilities of self-adaptation.

References

[1] Aarts, E., Korst, J.: Simulated Annealing and Boltzmann Machines, A Stochastic Approach to Combinatorial Optimization and Neural Computing, Chichester et al. 1989.

[2] Adams, J., Balas, E., Zawack, D.: The Shifting Bottleneck Procedure for Job Shop Scheduling, Management Science 34 (1988) 3, pp. 391-401.

[3] Barker, J.R., McMahon, G.B.: Scheduling the General Job-Shop, Management Science 31 (1985) 5, pp. 594-598.

[4] Carlier, J., Pinson, E.: An algorithm for solving the job-shop problem, Management Science 35 (1989) 2, pp. 164-176.

[5] Dorndorf, U., Pesch, E.: Evolution Based Learning in a Job Shop Scheduling Environment, Research Memorandum 92-019, University of Limburg, Limburg 1992.

[6] Dueck, G.: New Optimization Heuristics - The Great Deluge Algorithm and the Record-to-Record Travel, Journal of Computational Physics (1993) 104, pp. 86-92.

[7] Dueck, G., Scheuer, T.: Threshold Accepting: A General Purpose Optimization Algorithm Superior to Simulated Annealing, Journal of Computational Physics (1990) 90, pp. 161-175.

[8] Fang, H.-L., Ross, P., Corne, D.: A Promising Genetic Algorithm to Job-Shop Scheduling, Rescheduling, and Open-Shop Scheduling Problems, in: Forrest, S. (ed.): Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo 1993, pp. 375-382.

[9] Fisher, H., Thompson, G.L.: Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules, in: Muth, J.F., Thompson, G.L. (eds.): Industrial Scheduling, Englewood Cliffs 1963, pp. 225-251.

[10] Giffler, B., Thompson, G.L.: Algorithms for Solving Production-Scheduling Problems, Operations Research 8 (1960), pp. 487-503.

[11] Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning, Reading et al. 1989.

[12] Nakano, R., Yamada, T.: Conventional Genetic Algorithm for Job Shop Problems, in: Belew, R., Booker, L. (eds.): Proceedings of the Fourth International Conference on Genetic Algorithms, San Mateo 1991, pp. 474-479.

[13] Siedentopf, J.: An Efficient Scheduling Algorithm Based upon Threshold Accepting, Report No. 16 of the Institute of Production Management and Industrial Information Management, University of Leipzig, Leipzig 1995.

[14] Van Laarhoven, P.J.M., Aarts, E.H.L., Lenstra, J.K.: Job Shop Scheduling by Simulated Annealing, Operations Research 40 (1992) 1, pp. 113-125.

UNIVERSITY OF LEIPZIG

Institute of Production Management and Industrial Information Management

- Reports -

No. 1: Zelewski, Stephan: Das Konzept technologischer Theorietransformationen - eine Analyse aus produktionswirtschaftlicher Perspektive, Leipzig 1994.

No. 2: Siedentopf, Jukka: Anwendung und Beurteilung heuristischer Verbesserungsverfahren für die Maschinenbelegungsplanung - Ein exemplarischer Vergleich zwischen Neuronalen Netzwerken, Simulated Annealing und genetischen Algorithmen, Leipzig 1994.

No. 3: Zelewski, Stephan: Unternehmenskrisen und Konzepte zu ihrer Bewältigung, Leipzig 1994.

No. 4: Siedentopf, Jukka: Ein effizienter Scheduling-Algorithmus auf Basis des Threshold Accepting, Leipzig 1995.

No. 5: Zelewski, Stephan: Petrinetzbasierte Modellierung komplexer Produktionssysteme (Projekt PEMOPS), Band 1: Exposition, Leipzig 1995.

No. 6: Zelewski, Stephan: Petrinetzbasierte Modellierung komplexer Produktionssysteme (Projekt PEMOPS), Band 2: Bezugsrahmen, Leipzig 1995.

No. 7: Zelewski, Stephan: Petrinetzbasierte Modellierung komplexer Produktionssysteme (Projekt PEMOPS), Band 3: Einführung in Stelle/Transition-Netze, Leipzig 1995.

No. 8: Zelewski, Stephan: Petrinetzbasierte Modellierung komplexer Produktionssysteme (Projekt PEMOPS), Band 4: Verfeinerungen von Stelle/Transition-Netzen, Leipzig 1995.

No. 9: Zelewski, Stephan: Petrinetzbasierte Modellierung komplexer Produktionssysteme (Projekt PEMOPS), Band 5: Einführung in Synthetische Netze, Teilband 5.1: Darstellung des Kernkonzepts, Leipzig 1995.

No. 10: Zelewski, Stephan: Petrinetzbasierte Modellierung komplexer Produktionssysteme (Projekt PEMOPS), Band 5: Einführung in Synthetische Netze, Teilband 5.2: Auswertungsmöglichkeiten, Leipzig 1995.

No. 11: Zelewski, Stephan: Petrinetzbasierte Modellierung komplexer Produktionssysteme (Projekt PEMOPS), Band 6: Erweiterungen von Synthetischen Netzen, Leipzig 1995.

No. 12: Zelewski, Stephan: Petrinetzbasierte Modellierung komplexer Produktionssysteme (Projekt PEMOPS), Band 7: Fallstudie, Leipzig 1995.

No. 13: Zelewski, Stephan: Petrinetzbasierte Modellierung komplexer Produktionssysteme (Projekt PEMOPS), Band 8: Charakterisierung des Petrinetz-Konzepts, Leipzig 1995.

No. 14: Zelewski, Stephan: Petrinetzbasierte Modellierung komplexer Produktionssysteme (Projekt PEMOPS), Band 9: Beurteilung des Petrinetz-Konzepts, Leipzig 1995.

No. 15: Zelewski, Stephan: Petrinetzbasierte Modellierung komplexer Produktionssysteme (Projekt PEMOPS), Band 10: Petrinetz-Literatur, Leipzig 1995.

No. 16: Siedentopf, Jukka: An Efficient Scheduling Algorithm Based upon Threshold Accepting, Leipzig 1995.

No. 17: Siedentopf, Jukka: The Threshold Waving Algorithm for Job Shop Scheduling, Leipzig 1995.