RESEARCH PAPERS OF WROCŁAW UNIVERSITY OF ECONOMICS No. 147
Advanced Information Technologies for Management – AITM 2010 2010

Radosław Rudek
Wrocław University of Economics, Wrocław, Poland
e-mail: radoslaw.rudek@ue.wroc.pl

METAHEURISTIC ALGORITHMS FOR OPTIMIZATION OF VARYING EFFICIENCY PRODUCTION SYSTEMS

Abstract: In this paper, we analyse a schedule management problem in production systems with varying efficiency caused by the learning effect. The measurable result of learning is that the time required to produce a single item decreases as more units are produced. First, we provide a short historical review on the discussed issue and present different approaches to model the learning effect in the scheduling context. Since the considered problem seems to be NP-hard, we design and implement fast heuristics and metaheuristic algorithms which are based on simulated annealing, tabu search and genetic algorithms. Numerical evaluation of their efficiency is provided.

Keywords: scheduling, learning effect, heuristic.

1. Introduction

In practice the efficiency of manufacturing, industrial and other real-life systems is seldom constant since it is often affected by the learning effect. The measurable result of learning is that time required to produce a single item decreases as more similar units are produced. For the first time a quantitative relation between learning variables was described by Wright in 1936 on the examples known from the aircraft industry [Wright 1936]. He stated that the unit processing time decreases 20% with every redoubling of the output, that is called the 80% hypothesis. In general, the Wright's function, called *learning curve*, describes the time $p(v)$ required to produce the v-th unit of the same product as follows:

$$p(v) = pv^{\alpha}, \tag{1}$$

where p is the time required to produce the first unit and α is a learning index (slope of the learning curve) that depends on the learning rate LR, i.e., $\alpha = \log_2 LR$. The learning rate is defined as the rate of each redoubling the output, i.e., $LR = p(2v)/p(v)$. For the

80% hypothesis the learning index is calculated as follows $\alpha = \log_2 0.8 = -0.322$. The function (1) is often called log-linear learning curve.

One of the main factors that persuaded practitioners to the analysis of learning in aircraft industry was the high cost of airplanes [Kerzner 1998]. Furthermore, in the 1940s USA War Production Board used the methodology proposed by Wright to estimate the number of airplanes that can be produce for a given complement of men and machines [Roberts 1983].

The other studies on the learning effect proved its significant impact on productivity in manufacturing systems specialized in Hi-Tech electronic equipment [Adler, Clark 1991], memory chips and circuit boards [Webb 1994], electronic guidance systems [Kerzner 1998] and in many others (e.g. [Carlson, Rowe 1976; Cochran 1960; Holzer, Riahi-Belkaoui 1986; Jaber, Bonney 1999; Lien, Rasch 2001; Yelle 1979]).

However, the investigations also revealed that some systems are more precisely described by other characteristics (learning curves), e.g., S-shaped [Jaber, Bonney 1999], Stanford-B [Garg, Milliman 1961] or DeJong [Dejong 1957]. For more details concerning different learning curves see also [Holzer, Riahi-Belkaoui 1986; Lien, Rasch 2001; Yelle 1979].

The concept proposed by Wright allows to estimate the time parameters of a production changed by learning, however, it does not take into consideration the utilization of learning to improve system performances. It follows from the fact that (1) assumes the processing time p is identical for each product. Nevertheless, in many manufacturing and industrial systems the produced items are similar but not identical. It was emphasized by Biskup [1999], who assumed that for the given group of n items to be produced each of them can be different (however similar). Therefore based on (1), the time $p_j(v)$ required to processed job (product) j ($j = 1, ..., n$) if it is performed as the v-th in a production sequence is given as follows:

$$p_j(v) = p_j v^{\alpha}, \quad j = 1, ..., n, v = 1, ..., n, \tag{2}$$

where p_j is the normal processing time of job (item) j if it is processed as the first one (i.e., a human worker is not learnt to produce the given group of items).

Model (2) revealed that production time-objectives can be managed by sequence (schedule) of produced items by rational utilization of the learning effect. Instantly this direction of research has attracted particular attention of operational research and decision science community and find many followers in scheduling domain. It results from the fact that the application of learning models to describe industrial processes and on their basis to construct efficient management strategies can improve a production schedule and increase productivity (i.e., profit).

Although model (2) has a strong practical background (see [Yelle 1979]) it neglects the processing times of already performed jobs. Therefore, if human interactions have a significant impact on job processing times then the processing

time of each performed job is added to the worker experience and improves his efficiency [Biskup 2008]. Such real-life settings are covered by the model proposed by Kuo and Yang [2006], where the processing time of job j if it processed as the v-th in a sequence is given as follows:

$$p_j(v) = p_j\left(1 + \sum_{l=1}^{v-1} p_{[l]}\right)^{\alpha}, \tag{3}$$

where $\sum_{l=1}^{v-1} p_{[l]}$ is the sum of the normal processing times of jobs preceded job j and $p_{[l]}$ is the normal processing time of a job scheduled in the l-th position in a sequence.

Whereas model (2) assumes that learning is a result of processing independent operations like setting up machines, model (3) takes into consideration that experience gained by the worker is related with the normal (nominal) time required to process a job. To jobs represented by this model belong: offset printing (running the press itself is a highly complicated and error-prone process), assembling highly customized products, the production of high-end electric tools, maintenance of airplanes, pimping cars, etc. [Biskup 2008]. For survey on scheduling problems with the learning effect see [Biskup 2008] and [Janiak, Rudek 2009].

For the practical reasons production scheduling usually focus on the minimization of the following time-objectives: the maximum completion time, the maximum lateness, the sum of the weighted completion times, the number of late jobs and the sum of the weighted tardiness. In this paper, we focus on determining a schedule of jobs for a processor that learns according to model (3) and the objective is to minimize the sum of the weighted completion times. Since the considered problem seems to be NP-hard, it is highly unlikely to find an optimal algorithm that solves it in a reasonable time. Therefore, in this paper, we focus on designing and implementation of approximation methods, i.e., fast heuristics and first and foremost metaheuristic algorithms that are based on simulated annealing [Kirkpatrick et al. 1983], tabu search [Glover, Laguna 1997] and genetic algorithms [Holland 1975; Michalewicz 1996].

The remainder of this paper is organized as follows. The next section contains problem formulation. In Section 3 solution algorithms are presented, whereas the numerical verification of their efficiency is provided subsequently. The last section concludes the paper.

2. Problem formulation

In this section, we express the discussed production problem in the scheduling context. A human worker is identified with a *processor* and the items (e.g., raw materials, semi-finished products) which he has to process (e.g., to machine or to assemble a final product) are called *jobs*.

Let $J = \{1, ..., j, ..., n\}$ denote the set of n jobs that have to be performed by the processor. By the practical reason it is assumed the processor performs jobs without preemptions, otherwise for instance a renewed calibration of a lathe machine is required or even a semi-finished product can be damaged. We also assume that there are no precedence constraints between jobs, e.g., the order of machining raw materials on a lathe or processing semi-finished product can be performed in an arbitrary order. Moreover, we assume that jobs are continuously available for processing (e.g., machining, assembling). Finally, each job j is characterized by its weight parameter w_j (priority) and the function (learning curve) $p_j(v)$ describing its processing time according to (3).

Let $\pi = \langle \pi(1), \pi(2), ..., \pi(i), ..., \pi(n) \rangle$ denote the sequence/schedule of jobs (i.e., permutation of the elements of the set J), where $\pi(i)$ is the index of a job processed in position i in this sequence. By Π we will denote the set of all such permutations. For the given sequence (permutation) $\pi \in \Pi$, we can easily determine the completion time $C_{\pi(i)}$ of a job placed in the i-th position in π from the following formulae:

$$C_{\pi(i)} = \sum_{l=1}^{i} p_{\pi(l)}(l) = \sum_{l=1}^{i} p_{\pi(l)} \left(1 + \sum_{k=1}^{l-1} p_{\pi(k)} \right)^{\alpha}. \tag{4}$$

On this basis, we formulate the sum of weighted completion times (the total weighted completion times) being a function of the schedule π:

$$TWC(\pi) = \sum_{i=1}^{n} w_{\pi(i)} C_{\pi(i)}. \tag{5}$$

The objective is to find such a schedule (i.e., sequence) π of jobs performed by the processor that minimizes the total weighted completion time criterion (5). Formally the optimal schedule $\pi^* \in \Pi$ for the considered minimization objective is defined as follows

$$\pi^* = \arg\min_{\pi \in \Pi} \{TWC(\pi)\}. \tag{6}$$

For convenience, we will denote the considered problem by *TWC*.

In the further part of this paper, we will provide algorithms that solve the formulated problem.

3. Solution algorithms

The considered problem seems to be NP-hard, however, no formal proof has been provided in the scientific literature. Therefore, it is highly unlikely to find an optimal algorithm for the problem that solves it in a reasonable time for the number of jobs that occurs in practice (i.e., hundreds). Hence in this section, we construct approximation algorithms with low computational complexity. Namely, fast heuristics and

metaheuristic algorithms which are based on the popular and efficient optimization methods: simulated annealing [Kirkpatrick et al. 1983], tabu search [Glover, Laguna 1997] and genetic algorithms [Holland 1975; Michalewicz 1996].

3.1. Heuristics SWPT and NEH

First, we define a construction algorithm called Shortest Weighted Processing Times (SWPT), since it schedules the jobs according to the non-decreasing order of their coefficient p_j/w_j. SWPT was proved in [Jackson 1955] to be optimal for a problem with constant job processing times that from the perspective of this paper is a special case of the problem TWC with constant job processing times (i.e., $\alpha = 0$). The complexity of SWPT is $O(n \log n)$.

The second algorithm is based on the NEH algorithm presented in [Nawaz et al. 1983]. Its complexity is $O(n^3)$. Formal definition of NEH is presented in Figure 1.

1: DETERMINE THE INITIAL SEQUENCE OF JOBS IN $\pi_{initial}$ AND SET $\pi^* = \emptyset$
2: GET THE FIRST JOB j FROM $\pi_{initial}$
3: INSERT j IN SUCH A POSITION IN π^* FOR WHICH $TWC(\pi^*)$ IS MINIMAL
4: REMOVE j FROM $\pi_{initial}$
5: IF $\pi_{initial} \neq \emptyset$ GO TO STEP 2
6: THE PERMUTATION π^* IS THE GIVEN SOLUTION

Figure 1. NEH algorithm

Source: based on [Nawaz et al. 1983].

3.2. Simulated annealing

Simulated annealing (SA) [Kirkpatrick et al. 1983] is based on the metallurgical process. In the given implementation of SA let TWC(π)is the criterion value for the permutation π. The algorithm starts with an initial solution π_{initial} and generates in each iteration i a new permutation π' that is based on the current solution π by interchanging of two random jobs in π. This new solution π' replaces π (i.e., $\pi = \pi'$) with the following probability

$$P(T, \pi', \pi) = \min\left\{1, \exp\left(-\frac{TWC(\pi') - TWC(\pi)}{T}\right)\right\}, \tag{7}$$

where T is the temperature that decreases in a logarithmical manner

$$T = \frac{T}{1+\lambda T}, \tag{8}$$

and the values of the initial temperature T_0 and of the parameter λ are chosen empirically. The stopping condition of SA is the number of *Iterations*, thus, its overall computational complexity is $O(Iterations \cdot n)$. The formal definition of SA is given in Figure 2.

1: $T = T_0$, $\pi = \pi^* = \pi_{initial}$
2: FOR $i = 1$ TO $Iterations$
3: CHOOSE π' BY A RANDOM INTERCHANGE OF TWO JOBS IN π
4: ASSIGN $\pi = \pi'$ WITH PROBABILITY

$$P(T, \pi', \pi) = \min\left\{1, exp\left(-\frac{TWC(\pi') - TWC(\pi)}{T}\right)\right\}$$

5: IF $TWC(\pi) < TWC(\pi^*)$ THEN $\pi^* = \pi$
6: $T = \frac{T}{1+\lambda T}$
7: THE PERMUTATION π^* IS THE GIVEN SOLUTION

Figure 2. Simulated annealing (SA) based on

Source: [Kirkpatrick et al. 1983].

3.3. Tabu search

Tabu search (TS) algorithm [Glover, Laguna 1997] uses local search with a short term memory, called *tabu list*, which stores forbidden moves, hence allows TS to escape from local minima. In the implemented algorithm *move* is defined as the insertion of a job being in position j in a sequence into position v. The applied tabu list stores pairs (j, v) of forbidden moves. If (j, v) is in the tabu list, then for any sequence the insertion of a job from position j into position v is forbidden. The tabu list is organized as FIFO (First In First Out), thereby, if the list is full then the new pair (j, v) is added at its beginning overriding the previous pair occupying this position. The computational complexity of the applied TS is $O(Iterations \cdot (|TabuList| + n) \cdot n^2)$, where $|TabuList|$ is the size of the tabu list. The formal definition of TS is given in Figure 3.

```
1:  TabuList = ∅, π = π_best = π* = π_initial,
    TWC_best = TWC* = TWC(π*)
2:  For i = 1 to Iterations
3:    For j = n to 1
4:      For v = n to 1
5:        π' = π, Insert π'(j) to v in π'
6:        If j ≠ v and TWC(π') < TWC_best
7:          If (j,v) is not in TabuList
8:            π_best = π', TWC_best = TWC(π'),
              j_best = j, v_best = v
9:    Assign π = π_best
10:   Add (j_best, v_best) to TabuList
11:   If TWC_best < TWC*
12:     π* = π_best, TWC* = TWC_best
13: The permutation π* is the best found solution
```

Figure 3. Tabu search (TS)

Source: based on [Glover, Laguna 1997].

3.4. Genetic algorithm

Genetic algorithms (GA) [Holland 1975] are inspired by nature, namely evolution. GA starts with a set of initial solutions $\Pi_{population}$ called *population*, from which a subset $\Pi_{parents}$ is chosen that is called *parents*. The solutions from the set parents are crossed to obtain a new set of solution $\Pi_{children}$ called *children*. The solutions from this set can be further changed with probability $P_{mutation}$ that is called *mutation*. The last step is to select a new population (solutions) of size $|\Pi_{population}|$ from the set $\Pi_{parents} \cup \Pi_{children}$.

In the considered implementation of GA the initial population is provided by swap operations on the initial permutation $\pi_{initial}$. The set of parents is drawn from population and a new population (from the old population and children) is chosen according to the greedy strategy that select solutions with the lowest criterion values. Mutation is implemented as a swapping of two random jobs. The best found solution is always in $\Pi_{population}$. The computational complexity of the implemented GA is $O(Iterations \cdot (|\Pi_{population}| + |\Pi_{children}|) \cdot n)$. Formal definition of GA is given in Figure 4.

```
1: INITIAL POPULATION Π_population , π* = π_initial , C_max = C_π*
2: FOR i = 1 TO Iterations
3:    Π_parents = DRAW |Π_parents| PARENTS FROM Π_population ,
4:    Π_children = CROSSOVER OX ON CONSECUTIVE PAIRS OF Π_parents
5:    MUTATE Π_children WITH PROBABILITY p_mutation
6:    Π_population = CHOOSE |Π_population| BEST FROM Π_population ∪ Π_children
7:    π* = arg min_{π∈Π_population∪π*} {TWC(π)}
8: THE PERMUTATION π* IS THE BEST FOUND SOLUTION
```

Figure 4. Genetic algorithm (GA)

Source: based on [Holland 1975].

4. Numerical experiments

The aim of the numerical experiments is to verify the effectiveness of the proposed algorithms. The algorithms were coded in C++ and simulations were run on PC, Intel® Core™2 Duo Processor E4500 and 2GB RAM.

The proposed algorithms are evaluated for the following problem sizes $n = \{10, 25, 50, 100\}$. For each value of n, 100 random instances were generated from the uniform distribution in the following ranges of parameters: $p_j \in [1, 10]$ (or $p_j \in [1, 20]$) and $w_j \in [1, 5]$; for all jobs $\alpha = -0.322$.

Values of the parameters of the algorithms are chosen empirically as follows:

– (SA) *Iterations* = 100 000, T_0 = 1 000 000 and $\lambda = 0.01$;
– (TS) *Iterations* = 50, |*TabuList*| = 5;
– (GA) *Iterations* = 200, $|\Pi_{population}| = 160$, $|\Pi_{parents}| = 120$ and $P_{mutation} = 0.005$.

The initial solution for SA, TS and GA is random (i.e., a natural permutation), but if it is provided by NEH algorithm (which initial permutation is determined by SWPT), then these algorithms are denoted by SAN, TSN and GAN, respectively.

Each algorithm $A \in$ {SWPT, NEH, SA, SAN, TS, TSN, GA, GAN} is evaluated – for each instance I – according to the relative error $\delta_A(I)$ that is calculated in the following way:

$$\delta_A(I) = \frac{TWC(\pi_I^A) - TWC(\pi_I^*)}{TWC(\pi_I^*)} \cdot 100\%, \tag{9}$$

where $TWC(\pi_I^A)$ denotes the criterion value provided by algorithm A for instance I and $TWC(\pi_I^*)$ is the optimal solution (if $n = 10$) or the best known solution (if $n = \{25, 50, 100\}$) of instance I provided by the considered algorithms.

The results concerning mean and maximum relative errors provided by the analysed algorithms and their mean running times are presented in Tables 1 and 2, respectively.

Table 1. Mean and maximum (in square brackets) relative errors of algorithms and number of instances (in round brackets) for which each algorithm provided the best criterion value (the best results are emphasized)

n	p_j	SWPT	NEH	SA	SAN	TS	TSN	GA	GAN
10	[1, 10]	1.07 [8.33] (40)	0.45 [6.36] (62)	**0** **[0]** **(100)**	**0** **[0]** **(100)**	**0** **[0]** **(100)**	**0.00** **[0.00]** **(100)**	10.77 [28.84] (0)	0.45 [6.36] (62)
	[1, 20]	1.35 [13.23] (29)	0.73 [13.23] (53)	**0** **[0]** **(100)**	**0** **[0]** **(100)**	**0** **[0]** **(100)**	**0.00** **[0.00]** **(100)**	11.75 [32.09] (0)	0.61 [5.64] (53)
25	[1, 10]	0.33 [1.71] (5)	0.15 [1.70] (27)	**0** **[0]** **(100)**	**0** **[0]** **(100)**	**0** **[0]** **(100)**	**0.00** **[0.00]** **(100)**	27.70 [45.78] (0)	0.15 [1.70] (27)
	[1, 20]	0.47 [4.54] (1)	0.25 [4.31] (10)	**0** **[0]** **(100)**	**0** **[0]** **(100)**	**0** **[0]** **(100)**	**0.00** **[0.00]** **(100)**	30.80 [71.14] (0)	0.25 [4.31] (10)
50	[1, 10]	0.13 [0.66] (0)	0.06 [0.52] (9)	<0.001 [0.01] (85)	<0.001 [0.01] (86)	**0** **[0]** **(100)**	**0.00** **[0.00]** **(100)**	35.81 [58.17] (0)	0.06 [0.52] (9)
	[1, 20]	0.15 [0.74] (0)	0.09 [0.74] (3)	<0.001 [0.01] (81)	<0.001 [0.01] (85)	<0.001 [0.01] (94)	**0.00** **[0.00]** **(100)**	40.22 [57.73] (0)	0.09 [0.74] (3)
75	[1, 10]	0.06 [0.53] (0)	0.02 [0.13] (23)	0.011 [0.04] (5)	0.007 [0.04] (23)	0.58 [1.34] (0)	**0.00** **[0.00]** **(100)**	39.44 [53.99] (0)	0.02 [0.13] (23)
	[1, 20]	0.09 [0.66] (0)	0.04 [0.34] (1)	0.017 [0.04] (3)	0.012 [0.04] (3)	0.60 [1.55] (0)	**0.00** **[0.00]** **(100)**	43.90 [59.48] (0)	0.04 [0.34] (1)
100	[1, 10]	0.04 [0.17] (0)	0.01 [0.15] (23)	0.03 [0.10] (0)	0.01 [0.05] (23)	2.78 [4.76] (0)	**0.00** **[0.00]** **(100)**	42.17 [52.29] (0)	0.01 [0.15] (23)
	[1, 20]	0.06 [0.26] (0)	0.03 [0.26] (0)	0.04 [0.10] (0)	0.02 [0.07] (0)	2.94 [5.82] (0)	**0.00** **[0.00]** **(100)**	46.24 [57.85] (0)	0.03 [0.26] (0)

Table 2. Mean running times of the algorithms [s] (<0.001 means that the measured values are lower than 1 ms)

n	SWPT	NEH	SA	TS	GA
10	<0.001	<0.001	0.77	**0.04**	0.81
25	<0.001	<0.001	1.43	**0.45**	1.57
50	<0.001	0.02	**2.52**	3.18	3.38
75	<0.001	0.06	**3.59**	10.23	5.87
100	<0.001	0.14	**4.64**	23.54	9.05

The results presented in Tables 1 and 2 revealed the constructed heuristics provide solutions with very low relative error. It is worth noticing the good performances of SWPT, which maximal error does not exceed 13.5% and mean is lower than 1.4%. It constitutes an initial solution for NEH and metaheuristics.

On the other hand among the algorithms SA, TS, GA that start with natural permutation as the initial solution the lowest relative errors are provided by TS if n is not greater than 50. However, for greater instances SA is better, since having *Iterations* = 100 000, it escapes from local minima more efficiently than TS with relatively low number of iterations, i.e., *Iterations* = 50.

The influence of relatively good initial solutions (provided by SWPT and NEH) for SA is first of all visible in the number of instances for which SAN reach the best found solution. However, the influence of good initial solutions is crucial for TS, since it allows TS for a better start. It can be observed that TSN has found the best solutions in the all experiments, whereas SAN for n = 100 reach the best criterion only 23 times in 100 simulations. The worst results are received by GA and GAN only few times improved the initial results provided by SWPT+NEH. Thereby, the presented implementation of GA is inefficient for the considered problem.

The lowest running time characterizes TS for n not greater than 25 and then SA is even 5 times better (for n = 100). It results strictly from the complexity of these algorithms that is determined in Section 3.

Finally, the values of p_j have a marginal impact on the relative errors and they do not affect the running times of algorithms.

5. Conclusions

In this paper, we constructed, implemented and evaluated heuristic and metaheuristic solution algorithms for the minimization of the sum of the weighted completion times with the learning effect.

The numerical analysis revealed that the proposed algorithms are characterized with very low relative error in reference to the optimal solution that did not exceed 13.5% for the simplest SWPT and it was always optimal for the best algorithm TSN. The running times of the algorithms are reasonable and even for the most

efficient TSN they were about 24 s for 100 jobs. The experiments also showed that the presented implementation of GA is inefficient for the considered problem.

Our future work will focus on decreasing the complexity of NEH and TS as well as on the improvement of GA.

References

Adler P.S., Clark K.B. (1991), Behind the learning curve: A sketch of the learning process, *Management Science*, Vol. 37, pp. 267-281.

Biskup D. (1999), Single-machine scheduling with learning considerations, *European Journal of Operational Research*, Vol. 115, pp. 173-178.

Biskup D. (2008), A state-of-the-art review on scheduling with learning effects, *European Journal of Operational Research*, Vol. 188, pp. 315-329.

Carlson J.G., Rowe R.G. (1976), How much does forgetting cost?, *Industrial Engineering*, Vol. 8, pp. 40-47.

Cochran E.B. (1960), New concepts of the learning curve, *The Journal of Industrial Engineering*, Vol. 11, pp. 317-327.

Dejong J.R. (1957), The effects of increasing skill on cycle time and its consequences for time standards, *Ergonomics*, Vol. 1, pp. 51-60.

Garg A., Milliman P. (1961), The aircraft progress curve modified for design changes, *Journal of Industrial Engineering*, Vol. 12, pp. 23-27.

Glover F., Laguna M. (1997), *Tabu Search*, Kluwer, Norwell, MA.

Holland J.H. (1975), *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, MI.

Holzer H.P., Riahi-Belkaoui A. (1986), *The Learning Curve: A Management Accounting Tool*, Quorum Books, Westport, CT.

Jaber Y.M., Bonney M. (1999), The economic manufacture/order quantity (EMQ/EOQ) and the learning curve: Past, present, and future, *International Journal of Production Economics*, Vol. 59, pp. 93-102.

Jackson J.R. (1955), *Scheduling a Production Line to Minimize Maximum Tardiness*, Research Report 43, Management Sciences Research Project, UCLA, Los Angeles.

Janiak A., Rudek R. (2009), Experience based approach to scheduling problems with the learning effect, *IEEE Transactions on Systems, Man, and Cybernetics – Part A*, Vol. 39, pp. 344-357.

Kerzner H. (1998), *Project Management: A System Approach to Planning, Scheduling, and Controlling*, John Wiley & Sons, New York.

Kirkpatrick S., Gelatt C.D., Vecchi M.P. (1983), Optimization by simulated annealing, *Science*, Vol. 220, pp. 671-680.

Kuo W.-H., Yang D.-L. (2006), Minimizing the total completion time in a single-machine scheduling problem with a time-dependent learning effect, *European Journal of Operational Research*, Vol. 174, pp. 1184-1190.

Lien T.K., Rasch F.O. (2001), Hybrid automatic-manual assembly systems, *Annals of the CIRP*, Vol. 50, pp. 21-24.

Michalewicz Z. (1996), *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, New York.

Nawaz M., Enscore E.E. Jr., Ham I.A. (1983), A heuristic algorithm for *m*-machine, *n*-jobs flow-shop sequencing problem, *OMEGA International Journal of Management Science*, Vol. 11, pp. 91-95.

Roberts P. (1983), A theory of the learning process, *Journal of the Operational Research Society*, Vol. 34, pp. 71-79.

Webb G.K. (1994), Integrated circuit (IC) pricing, *High Technology Management Research*, Vol. 5, pp. 247-260.

Wright T.P. (1936), Factors affecting the cost of airplanes, *Journal of Aeronautical Sciences*, Vol. 3, pp. 122-128.

Yelle L.E. (1979), The learning curve: Historical review and comprehensive study, *Decision Science*, Vol. 10, pp. 302-328.

ALGORYTMY METAHEURYSTYCZNE W OPTYMALIZACJI SYSTEMÓW PRODUKCYJNYCH O ZMIENNEJ EFEKTYWNOŚCI

Streszczenie: W pracy analizowano problem harmonogramowania występujący w systemach produkcyjnych o zmiennej efektywności spowodowanej efektem uczenia. Mierzalnym rezultatem tego zjawiska jest skrócenie czasów potrzebnych do wykonania określonych produktów. W pierwszej kolejności dokonano krótkiego przeglądu literatury dotyczącego problemów harmonogramowania z efektem uczenia. Następnie zaprojektowano szybkie metody heurystyczne i metaheurystyczne oparte na symulowanym wyżarzaniu, poszukiwaniu z zakazami oraz algorytmach genetycznych. Ponadto przeprowadzono analizę numeryczną ich efektywności.