Radosław Rudek, Agnieszka Wielgus

Metaheuristic algorithms for scheduling on parallel machines with variable setup times

Ekonomiczne Problemy Usług nr 122, 369-377

2016

Ekonomiczne Problemy Usług nr 122

ISSN: 1896-382X | www.wnus.edu.pl/pl/edu/
DOI: 10.18276/epu.2016.122-35 | strony: 371-379

RADOSŁAW RUDEK
Wrocław University of Economics

AGNIESZKA WIELGUS
Wrocław University of Technology

METAHEURISTIC ALGORITHMS FOR SCHEDULING ON PARALLEL MACHINES WITH VARIABLE SETUP TIMES

Abstract

In order to meet growing demands of the market modern manufacturing and service environments must offer an increasingly broad range of services or products as well as ensure their required amount and short lead times. It can be done by the application of universal machines or workers which are able to perform different tasks. On the other hand, human activity environments are often affected by learning. Therefore, in this paper, we analyse related problems, which can be expressed as the makespan minimization scheduling problem on identical parallel machines with variable setup times affected by learning of workers. To provide an efficient schedule, we propose metaheuristic algorithms. Their potential applicability is verified numerically.
Keywords: scheduling, parallel machines, setup times, learning.

Introduction

Modern manufacturing and service environments to meet growing demands of the market must offer an increasingly broad range of services or products, ensure their required amount and short lead times. However, this causes significant increase in the complexity of the management process (Pinedo, 2012). It leads to a reduction in the efficiency of available resources utilization (workers, machines, etc.). Therefore, the increase of manufacturing/service organization competitiveness is also associated with the use of efficient methods supporting the management, that

will be able to meet these changes. This is especially evident in project management and production, where a plan or a schedule is determined such that it optimizes some given criterion (Allahverdi, 2015; Biskup, 2008; Pinedo, 2012).

Often there is a need to provide multiple products or services on common resources such as universal CNC machines or workers being able to perform different jobs. Thus, such environments require setups related with preparations of universal machines to process different products (e.g., tool changes, programming, etc.) or rearrangements of working places.

During the analysis of models of scheduling problems, it can be observed that they simultaneously evolve to better describe the phenomena existing in real-life. In recent years some additional factors affecting setup times/costs have been taken into account, such as learning of a human worker (Kuo, Hsu, and Yang, 2011). Namely, typical human activity environments as well as automatized manufacturing require human support for machines, which is needed during activities such as operating, controlling, setup, cleaning, maintaining, failure removal, etc. Usually, humans increase their efficiency with the number of repetitions of the same or similar activities (Biskup, 2008). It can result in decreasing of processing times of setups (Allahverdi, 2015). This phenomena of learning can be illustrated by Figure 1.

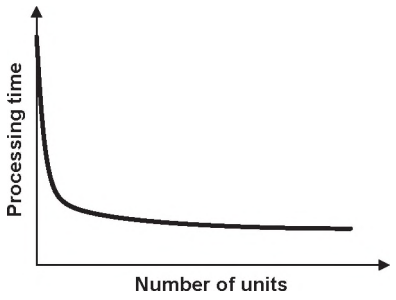

Fig. 1. Setup processing times depending on the number of performed setups (units)

Source: Following (Biskup 2008).

Therefore, in this paper, we will analyse such problems that can be expressed as scheduling on parallel machines (e.g., human workers or CNC machines) with setup times of machines under minimization of the maximum completion time (makespan). Additionally, we will take into consideration that setup times can be affected by learning of human workers. To solve the considered problem, we will propose metaheuristic algorithms that are based on a simulated annlealing technique.

The remainder of this paper is organized as follows. The problem is formulated in the next section, whereas its solution algorithms are given subsequently, followed by their numerical analysis. The last section concludes the paper.

1. Problem formulation

There is given a set $J = \{1,... ,n\}$ of n jobs (tasks) to be completed by a set $M = \{M_1,.., M_m\}$ of m identical machines (e.g., workers or CNC machines). It is assumed that jobs and machines are available at time zero. We assume that job preemption is not allowed and no precedence constraints between jobs exist. Moreover, each job is characterized by is processing time p_j.

Following practical cases, jobs can belong to different families. It means that if they are from the same family, they require the same sets of production facilities (e.g., tools, working place arrangement, etc.). However, a rearranging (setup) of a machine (preparation of tools, working place) is needed whenever there occurs a switching between jobs belonging to different families, which on the other hand is related with additional time. Let $F = \{I_f: f = 1, ..., b\}$ be a set of b families. Each job $j \in J$ belongs to only one family $I_f \in F$. Namely, if job j belongs to family I_f, then it is denoted by $g_j = f$. Thereby, if job k is processed just before job j on the same machine M_i and k belongs to a different family than j, i.e., $g_k \neq g_j$, then a setup of machine M_i is required. In other words, a preparation of a machine is needed to start job j, where $g_j = f$. It is related with a setup time s_f related with family I_f. If job j is performed as the first job on machine M_i, then a setup related with its family is needed. In this paper, we consider setups that are sequence independent, thereby a setup of a family does not depend on the previously performed family (Allahverdi, 2015).

In many real-life cases, setup times are not constant values, but they can vary due to the learning effect. It means that preparing a machine (or working place) to process jobs belonging to the same family is less time consuming if such preparations were done before. Following (Biskup, 2008), the setup time of family $s_f(v_i)$ is described by a non-decreasing function dependent on the number v_i of setups related with family I_f performed on machine (by worker) M_i (or by workers related with this machine) as follows:

$$s_f(v_i) = s_f v_i^{\alpha}, \tag{1}$$

where s_f is a normal setup time related with family I_f, if the setup is performed for the first time. The parameter α is a learning index (a slope of the learning curve) that depends on the learning rate LR ($\alpha = \log_2 LR$), which is defined as rate of each redoubling the output $LR = s_f(2v_i)/s_f(v_i)$. For the popular 80% hypothesis the learning index is calculated as follows $\alpha = \log_2 LR = -0.322$ (Biskup, 2008).

Let us now formulate the considered problem. At first, the schedule of jobs can be unambiguously defined by their permutations (sequences) $\pi = \{\pi_1, ..., \pi_i, ..., \pi_m\}$, where π_i denotes the permutation of jobs on machine M_i and $\pi_i(k)$ is the index of the kth job in permutation π_i for $i = 1,...,m$. Moreover, n_i is the number of jobs assigned to machine (worker) M_i. On this basis, the completion time of job $\pi_i(k)$, i.e., scheduled as the kth in permutation π on machine M_i, can be defined as follows:

$$C^{(i)}_{\pi_i(k)} = C^{(i)}_{\pi_i(k-1)} + p_{\pi_i(k)} + s_{\pi_i(k-1),\pi_i(k)} v_i^{\alpha}, \tag{2}$$

where $C^{(i)}_{\pi_i(0)} = 0$ (for $i = 1, ..., m$) and $s_{\pi_i(k-1),\pi_i(k)} = \begin{cases} s_{\pi_i(k)}, & g_{\pi_i(k)} \neq g_{\pi_i(k-1)} \\ 0, & otherwise \end{cases}$ is equal to the setup time $s_{\pi_i(k)}$ of job $\pi_i(k)$ if the previous job $\pi_i(k\text{-}1)$ belongs to a different family, otherwise $s_{\pi(k-1),\pi(k)} = 0$ if both jobs belong to the same family. Note that $s_{\pi_i(k)} = s_f$ if $g_{\pi_i(k)} = f$ and $s_{0,\pi_i(1)} = s_{\pi_i(1)}$ for $i = 1, ..., m$. Recall that the parameter v_i is the number of previous setups on machine M_i.

The objective is to find such a schedule of jobs π that minimizes the maximum completion time among all machines (makespan):

$$C_{\max}(\pi) = \max_{i=1,...,m} \{C^{(i)}_{\pi_i(n)}\}. \tag{3}$$

Thereby the problem can be formally defined as

$$\pi^* = \arg\min_{\pi \in \Pi} \{C_{\max}(\pi)\}, \tag{4}$$

where Π is the set of all possible schedules.

For convenience the problem will be used according to the standard three field notation scheme as follows Pm|SLE|C_{max} (i.e., Pm – parallel identical machines, SLE – Setup Learning Effect, C_{max} – the maximum completion time criterion, called the makespan).

2. Algorithms

The considered problem Pm|SLE|Cmax is at least NP-hard, since its classical version without setup times is at NP-hard (Pinedo, 2012). To solve it, we will propose some metaheuristic algorithms based on simulated annealing (Kirkpatrick, Gelatt, and Vecchi, 1983; Rudek, 2013).

In both algorithms, we use a representation of a solution, where a set of indices is used $\{\{1,..., n\}, \{n+1, ..., n+m-1\}\}$ such that indices $\{1,...,n\}$ refer to jobs, whereas $\{n+1,..., n+m-1\}$ are markers to separate jobs on particular machines. It is illustrated in the following example.

Example 1

Given $n = 6$ jobs and $m = 3$ machines. On this basis, the following set is constructed {{1, ...,6},{7, 8}}, where {1,...,6} are indices of jobs and {7,8} are used to separate them on particular machines. Therefore, a representation of a schedule $\pi = (1, 2, 3, 7, 4, 8, 5, 6)$, which is equivalent to (1, 2, 3, 8, 4, 7, 5, 6), refers to the following schedules of jobs on machines $\pi_1 = (1, 2, 3)$, $\pi_2 = (4)$, $\pi_3 = (5, 6)$.

The primary simulated annealing (SA) algorithm (e.g., Rudek, 2013) starts with an initial solution π_{init}. In each iteration of the algorithm, a new solution π_{new} is obtained on the basis of an old solution π_{old}. It is done by the interchanging of two randomly chosen jobs. This new solution is accepted with the probability

$$\max\left\{1, \exp\left(-\frac{C_{\max}(\pi_{new}) - C_{\max}(\pi_{old})}{T}\right)\right\}, \tag{5}$$

where $T=T/(1+\lambda)$ is a temperature parameter, which decreases in each iterations. The parameter λ is calculated as follows:

$$\lambda = \frac{T_0 - T_N}{NT_0T_N}, \tag{6}$$

where T_0 is an initial value of T, and T_N is its final value, whereas N is the assumed number of iterations of SA. The value of T_N is close to zero and the initial value of T_0 is calculated according to Algorithm 1.

Algorithm 1 Calculate initial value T_0

```
Cmin = ∞
Cmax = 0
for iter = 1 to n*n
{
    i = random value from 1...n
    j = random value from 1...n
    swap jobs in positions i and j in π
    Calculate criterion value C for π
    if C < Cmin then Cmin = C
    if C > Cmax then Cmax = C
}
return -(Cmax - Cmin)/log(0.9)
```

Finally, if a new solution π_{new} is better than the best already found π^*, i.e., $C_{\max}(\pi^*) > C_{\max}(\pi_{new})$, then $\pi^* = \pi_{new}$.

In this paper, we also extend the classical SA by a variable neighbourhood search (denoted by SAV), where swap and insert moves are applied. Its formal description is given as follows.

Algorithm 2 SAV

```
λ calculate according to (6)
T = T0 calculate according to Algorithm 1
πold = πnew = π* = πinit determine according to a list scheduling algorithm
while(stop condition not hold)
{
    for iter = 1 to IterMax
    {
       i = random value 1...n
       j = random value 1...n
       obtain πnew by swaping jobs in positions i and j in πold
       assign πold = πnew with probability defined by (5)
       if C(πnew) <C(π*) then π* = πnew
       obtain πnew by insertion of job from position i to j in πold
       assign πold = πnew with probability defined by (5)
       if C(πnew) <C(π*) then π* = πnew
    }
    T=T/(1+λ)
}
return schedule π*
```

Initial solution π_{init} of SA and SAV is provided by a list scheduling algorithm (LSA) that on the basis of a list (with randomly sequenced jobs) assigns jobs to the first available machine.

3. Numerical analysis

In this section, we will analyse numerically the algorithms described in the previous section.[1] During simulations, the following problem sizes were considered $n \in \{10, 100, 500\}$ and $m \in \{2, 5, 10\}$.

[1] All of them were coded in C++ and simulations were run on PC, CPU Intel® Core™ i7-2600K 3.40 GHz and 8 GB RAM.

Table 1

Mean and maximum relative errors of the analysed algorithms (running times 500ms)

n	m	b	s_j	LSA		SA		SAV	
				mean	max	mean	max	mean	max
10	2	[1,2]	[1, 5]	17.87	29.60	**0.00**	**0.00**	**0.00**	**0.00**
			[1,10]	25.63	48.01	**0.00**	**0.00**	**0.00**	**0.00**
			[1,20]	37.35	99.15	**0.00**	**0.00**	**0.00**	**0.00**
		[1,5]	[1, 5]	12.57	28.41	**0.00**	**0.00**	**0.00**	**0.00**
			[1,10]	28.46	52.73	**0.00**	**0.00**	**0.00**	**0.00**
			[1,20]	34.69	54.93	**0.00**	**0.00**	**0.00**	**0.00**
100	2	[1,10]	[1, 5]	16.69	21.86	0.47	2.32	**0.10**	**0.82**
			[1,10]	29.12	36.67	1.04	3.30	**0.16**	**1.45**
			[1,20]	58.97	78.21	**1.13**	**3.37**	1.38	6.39
		[1,50]	[1, 5]	11.84	16.15	**0.19**	**0.77**	0.21	0.83
			[1,10]	21.30	30.69	0.24	1.91	**0.13**	**0.67**
			[1,20]	37.53	43.50	0.44	1.57	**0.24**	**1.10**
	5	[1,10]	[1, 5]	20.48	24.90	**0.19**	**0.84**	0.45	2.54
			[1,10]	34.38	51.09	**0.23**	**1.03**	0.68	2.37
			[1,20]	71.04	90.69	1.55	7.30	**1.38**	**3.94**
		[1,50]	[1, 5]	14.26	20.22	0.34	2.59	**0.32**	**1.12**
			[1,10]	28.59	35.64	0.75	2.04	**0.54**	**1.89**
			[1,20]	45.66	62.76	0.76	2.65	**0.37**	**2.13**
	10	[1,10]	[1, 5]	21.97	28.58	0.71	2.70	**0.53**	**2.24**
			[1,10]	35.62	54.20	0.51	1.74	**0.30**	**1.68**
			[1,20]	67.40	101.41	0.98	4.76	**0.60**	**4.00**
		[1,50]	[1, 5]	17.22	30.23	0.33	1.34	**0.30**	**1.18**
			[1,10]	27.87	34.18	**0.19**	**1.64**	1.27	2.73
			[1,20]	51.88	68.37	2.25	4.59	**0.30**	**1.66**
500	2	[1,50]	[1, 5]	13.10	16.49	0.78	1.55	**0.00**	**0.02**
			[1,10]	24.49	30.22	1.25	2.78	**0.11**	**0.57**
			[1,20]	44.07	55.66	0.75	2.84	**0.30**	**1.57**
		[1,250]	[1, 5]	9.23	10.80	0.23	0.84	**0.05**	**0.29**
			[1,10]	15.33	18.33	0.56	1.41	**0.11**	**0.65**
			[1,20]	28.35	34.50	0.99	3.08	**0.21**	**1.49**
	5	[1,50]	[1, 5]	14.58	18.89	**0.07**	**0.56**	0.82	2.68
			[1,10]	26.08	33.40	**0.10**	**0.47**	0.63	1.89
			[1,20]	43.65	54.36	**0.27**	**1.80**	1.55	3.94
		[1,250]	[1, 5]	9.55	12.49	**0.07**	**0.41**	0.61	1.53
			[1,10]	16.63	22.07	**0.32**	**1.49**	0.54	1.60
			[1,20]	32.01	38.18	**0.22**	**1.15**	1.31	5.23
	10	[1,50]	[1, 5]	16.83	21.28	0.52	4.02	**0.47**	**1.22**
			[1,10]	25.13	37.19	**0.44**	**2.80**	1.10	5.47
			[1,20]	49.15	59.82	1.56	5.10	**0.88**	**4.14**
		[1,250]	[1, 5]	10.64	14.27	**0.16**	**0.46**	0.46	1.76
			[1,10]	18.68	27.96	**0.56**	**2.45**	0.67	2.82
			[1,20]	33.99	43.66	**0.41**	**2.66**	1.15	2.75

Source: own work.

For each pair of n and m, 100 different random instances were generated from the uniform distribution over the integers in the following ranges of parameters: the processing times $p_j \in \{1,..., 10\}$; the setup times $s_j \in \{1,...,5\}$, $\{1,...,10\}$, $\{1,..., 20\}$; the number of families $b \in \{0.1n, 0.5n\}$; the learning index $\alpha = -0.322$ that refers to a popular learning rate 80% (Biskup, 2008). The parameters of simulated annealing (SA and SAV) are chosen empirically as follows: $IterMax = 10$, $T_N = 0.0001$, $N = 280000n^{-0.7}$ and the stop condition is set to be 500 ms.

Similarly as in (Rudek, 2013), the algorithms are evaluated according to the relative error δ calculated as follows:

$$\delta^A(I) = \frac{C^A_{\max}(I) - C^*_{\max}(I)}{C^*_{\max}(I)} \times 100\%, \tag{7}$$

where $C^A_{\max}(I)$ is the criterion value obtained by algorithm $A \in \{\text{SA, SAV, TS}\}$ for instance I and $C^*_{\max}(I)$ is the optimal ($n = 10$) or best found criterion value for instance I. The mean and maximum relative errors of the algorithms are given in Table 1.

It can be seen in Table 1 that SAV provided similar results as SA. Moreover, the algorithms SA and SAV provided optimal solutions for all instances $n = 10$, and they are significantly better than LSA. It is worth mentioning that we have implemented and analysed different tabu search algorithms (based on insert and swap moves). However, they were overwhelmed by SAV and duo to page limit their descriptions as well as numerical analysis were omitted in this paper.

Conclusions

In this paper, we expressed some problems as the makespan minimization scheduling problem on identical parallel machines with non-increasing setup times dependent on the number of previous setups. We also proposed metaheuristic algorithms based on simulated annealing. The numerical analysis showed that the algorithms are efficient to solve the problem.

Our future work will focus on the analysis of parallel scheduling problems under other objectives as well as the construction of other metaheuristic algorithms.

Acknowledgement

The research presented in this paper has been partially supported by the Polish National Science Centre under grant no. DEC-2012/05/D/HS4/01129 (algorithms) and by the Polish Ministry of Science and Higher Education under Iuventus Plus Programme (No. IP2014 040673) (models/analysis).

Literature

1. Allahverdi, A. (2015). The third comprehensive survey on scheduling problems with setup times/costs. *European Journal of Operational Research*, 246, pp. 345–378.
2. Biskup, D. (2008). A state-of-the-art review on scheduling with learning effects. *European Journal of Operational Research*, 188, pp. 315–329.
3. Kirkpatrick, S. and Gelatt, C. D. and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220, pp. 671–680.
4. Kuo, W.-H. and Hsu, C.-J. and Yang, D.-L. (2011). Some unrelated parallel machine scheduling problems with past-sequence-dependent setup time and learning effects. *Computers & Industrial Engineering*, 61, pp. 179–183.
5. Pinedo, M. (2012). *Scheduling: Theory, Algorithms and Systems (4rd ed.)*. New York: Springer, 2012.
6. Rudek, R. (2013). On single processor scheduling problems with learning dependent on the number of processed jobs. *Applied Mathematical Modelling*, 37, pp. 1523–1536.

ALGORYTMY METAHEURYSTYCZNE DLA PROBLEMU HARMONOGRAMOWANIA ZADAŃ NA IDENTYCZNYCH MASZYNACH RÓWNOLEGŁYCH ZE ZMIENNYMI CZASAMI PRZEZBROJEŃ

Streszczenie

Współczesne przedsiębiorstwa przemysłowe i usługowe aby sprostać rosnącym wymaganiom rynkowym, muszą oferować coraz szerszy asortyment oferowanych usług lub produktów, a także zapewnić ich wymaganą ilość i szybkość realizacji. Można tego dokonać poprzez zastosowanie uniwersalnych maszyn lub pracowników, którzy potrafią realizować różne zadania. Z drugiej strony, istnienie czynnika ludzkiego powoduje występowanie efektu uczenia. W związku z tym w niniejszej pracy analizowany jest powiązany problem harmonogramowania na identycznych maszynach równoległych przy kryterium minimalizacji długości uszeregowania oraz przy zmiennych czasach przezbrojeń wynikających z efektu uczenia pracowników. W celu opracowania efektywnego harmonogramu zaproponowano algorytmy metaheurystyczne. Zakres ich zastosowania został zweryfikowany w oparciu o analizę numeryczną.
Słowa kluczowe: harmonogramowanie, maszyny równoległe, przezbrojenie, uczenie.

Tłumaczenie Radosław Rudek