Management and Production Engineering Review
Volume 10 • Number 3 • September 2019 • pp. 54–60
DOI: 10.24425/mper.2019.129598

EXPLORING HEURISTIC TECHNIQUES FOR FLOW SHOP SCHEDULING

Zuzana Soltysova[1], Pavol Semanco[2], Jan Modrak[3]

[1] *Faculty of Manufacturing Technologies, Technical University of Kosice, Slovakia*
[2] *Lear Corporation Seating Slovakia s.r.o., Slovakia*
[3] *T-systems, Slovakia s.r.o., Slovakia*

Corresponding author:
Zuzana Soltysova
Technical University of Kosice
Faculty of Manufacturing Technologies
Bayerova 1, 080 01 Presov, Slovakia
phone: (+421) 55-602-6402
e-mail: zuzana.soltysova@tuke.sk

Received: 7 September 2018
Accepted: 10 September 2019

Abstract
This paper explores selected heuristics methods, namely CDS, Palmer's slope index, Gupta's algorithm, and concurrent heuristic algorithm for minimizing the makespan in permutation flow shop scheduling problem. Its main scope is to explore how different instances sizes impact on performance variability. The computational experiment includes 12 of available benchmark data sets of 10 problems proposed by Taillard. The results are computed and presented in the form of relative percentage deviation, while outputs of the NEH algorithm were used as reference solutions for comparison purposes. Finally, pertinent findings are commented.

Keywords
Benchmarking, flow shop, makespan, relative percent deviation, heuristics.

Introduction

In this paper we focus on the permutation flow shop scheduling problem (PFSP) in manufacturing systems, where all jobs have to follow the same route in the same order and machines are assumed to be set up in series. We consider general permutation flow shop scheduling with unlimited intermediate storage where it is not allowed sequence changes between machines. Our attention is directed on permutation schedules with constant setup times that are included in processing times and to availability of all jobs at zero time. The general flow shop problem with a makespan ($C_{\max}$) objective can be denoted as $n/m/F/C_{\max}$ that involves n jobs where each requires operations on m machines, in the same job sequence. The solution of such problem is represented by the optimal job sequence that produces the smallest makespan assuming no preemption of jobs. Operations research literature recognizes two well-known types of heuristics: constructive and improvement heuristics, see, e.g., in [1, 2]. Johnson proposed the earliest known heuristic for the PFSP, which provides an optimal solution for two machines [3]. The computational complexity of Johnson's algorithm is $O(n \log n)$. Campbell, Dudek and Smith introduced a heuristic technique, which presents an extension of Johnson's algorithm [4]. Koulamas offered another modification of Johnson's Algorithm [5]. The technique consists of two phases. In the first phase, it makes extensive use of Johnson's algorithm, whereas the second one aims to improve the resulting schedule from the first phase by allowing job passing between machines. computational complexity of this heuristic is $O(m^2 n^2)$.

The improvement heuristics start with an initial solution and then provide a pattern for iteratively obtaining an improved solution [6]. These iterative approaches, referred to as meta-heuristic approaches, are inherently local search techniques such as, for example, tabu search (TS), simulated annealing (SA), genetic algorithms (GA), etc. Computational experiments presented in this paper compare results of four constructive heuristics, namely, the Campbell, Dudek and Smith (CDS) heuristic, Gupta's algorithm, Palmer's slope index, and modified Johnson's algorithm (CH) with reference values of NEH algorithm. We perform our study on datasets including 120 benchmark problems by [7]. The rest of the paper is organized as follows. Firstly, related literature review of the flow shop scheduling heuristics is briefly outlined. Next Section describes the algorithms used in this study. Subsequently decisive findings of computational experiments are summarized. Conclusion section articulates possible future research directions.

Related work

Flow shop scheduling problems that belong to the class of multi-stage scheduling problems are frequently encountered in the scheduling literature. The permutation flow shop problem is formulated as a mixed integer programming and it is classified as NP-Hard problem [8, 9]. Their attractiveness is due to their wide applications in practice, see, e.g. [11–14]. Makespan minimization is considered as one of the most meaningful objectives for flow shop production [15, 16]. Yenisey and Yagmahan provided a comprehensive review for all types of scheduling problems, including classification and current trends [17]. According to them, research in this domain has been focused mainly on single objectives over time. Nonetheless, in real conditions scheduling problems involve multiple objectives. The Multi-Objective PFSP problems are divided into three classes from viewpoint of the role of the decision maker in the process of production scheduling [18, 19]:

A priori approach – the decision maker at the decision making process provides all essential information at its begin in two possible ways: by minimization of a weighted combination of the objectives [20, 21]. And by hierarchically optimization of the objectives [22, 23].

A posteriori approach – the decision maker does a selection from the set of efficient solutions, which must be developed firstly [24–27]. *An interactive approach* – during the solution process, the preferences are presented by the decision maker, who interactively reflects the most preferred solution at each step of the procedure [28].

Yinga and Liao used the ACO approaches to get an optimal makespan in n-job and m-machine permutation flow shop [6]. Tran and Ng introduced a novel meta-heuristic called water flow-like algorithm [29]. Other related approaches are search methods like large neighborhood search [30, 31]. The latest works were focused e.g. on using a hybrid multi-objective backtracking search algorithm by [32] introducing a discrete differential evolution algorithm on the basis of an initial population generated by the upper bounds of the B&B algorithm by [33] proposing new heuristic tie-breaking rules in the implementation of NEH heuristic for PFSP by [34] solving the multi-objective distributed PFSP by competitive memetic algorithm by [35] proposing discrete artificial bee colony algorithm for multi-objective PFSP with sequence dependent setup times in [36], or introducing copula-based hybrid estimation of distribution algorithm for reentrant PFSP by [37].

Wider related work is included in recent review study on routing and scheduling [38].

Description of compared algorithms

In this section we formally explain the steps of the constructive heuristic approaches used to obtain a good initial solution. The following notations were used:

J set of n jobs $\{1, 2, ..., n\}$,

Mset of m machines $\{1, 2, ..., m\}$,

M_p set of two pseudo machines $\{1, 2\}$,

G set of 2 clusters $\{I, II\}$,

K number of k machines,

L number of $m - k$ machines,

I cluster of k machines,

$I\,I$ cluster of $m - k$ machines,

p_{ij} processing time of i-th job on j-th machine, $i \in J$ and $j \in M$,

P_j sum of processing time of n jobs on j-th machine,

$\sum P_g$ total sum of processing time of n jobs on machines in g-th group, $g \in G$,

it iteration number,

$it_{\max}$ max iteration number,

DIF_{it} difference between groups I and II,

$C_{\max}$ makespan,

C_j completion time,

s splitting ratio,

$s_{\max}$ max splitting ratio.

NEH algorithm

NEH is considered to be the best algorithm for the classic flow shop problem [39]. The NEH algorithm computes the sum of the processing times for each job and then lists them in non-increasing order of this value. The job at the top of the list is removed and inserted into the partial schedule. The position where it is inserted is determined by considering all the 8 possible positions it can occupy, without altering the relative positions of the jobs already in the partial schedule. The selected position is the one that minimizes the makespan in the partial schedule. This is repeated until the last of the unscheduled jobs is assigned.

CDS algorithm

This heuristic is a generalization of Johnson's two machine algorithm and it generates a set of $m-1$ artificial two-machine problems from an original m-machine problem, then each of the generated problems are solved using Johnson's algorithm [4].

The objective of the heuristic is the minimization of make-span in a deterministic flow shop problem. CDS heuristic forms in a simple manner a set of an $m-1$ artificial 2-machine sub-problems for the original m-machine problem by summing the processing times in a manner that combines $M1, M2, ..., M_{m-1}$ to pseudo machine 1 and $M2, M3, ..., M_m$ to pseudo machine 2.

Gupta's algorithm

Gupta, Hennig and Werner argued that the sequencing problem is a problem of sorting n items so as to minimize make-span [22]. He proposed algorithm to schedule sequence of jobs for more than two machines in a flow shop. Given a set of n independent jobs, each having m ($m > 2$) tasks that must be executed in the same sequence on m machines ($P1, P2, ..., P_m$). Output is a schedule with a minimum completion time of the last job.

Palmer's slope index

The heuristic has been developed in an effort to use Johnson's rule for $m \geq 3$, since for $m = 2$. This algorithm is slightly different from Johnson's algorithm. The idea of Slope Index method is to give priority to jobs so that jobs with processing times that tend to increase from machine to machine will receive higher priority, while jobs with processing times that tend to decrease from machine to machine will receive lower priority.

Modified Johnson's algorithm

This concurrent heuristic (CH) is using a difference between sums of processing times for each machine as a pair-splitting strategy to make two groups of the matrix of n-job and m-machine. Once the problem is converted to n job and 2 machines the sequence is determined using Johnson's algorithm. This approach can be described as follows [10]:

Step 1. Calculate the sums of processing times P_j of jobs on each machine.

Step 2. Split n-job and m-machine matrix and compute total sum of processing time. Split $n \times m$ matrix according to relation:

$$\sum_{j=1}^{k} P_I \sim \sum_{j=k-1}^{m} P_{II}. \tag{1}$$

Calculate $\sum P_I$, $\sum P_{II}$ of cluster I and II as follows:

$$\sum P_I = \sum_{j=1}^{k} P_j \forall k \in I, \ \ j \in M, \tag{2}$$

$$\sum P_{II} = \sum_{j=k-1}^{m} P_j \forall k \in I, \ \ j \in M. \tag{3}$$

Step 3. Compute the splitting ratio for this iteration that is given by:

$$s_{it} = \frac{\min(\sum P_I;\ \sum P_{II})}{\max(\sum P_I;\ \sum P_{II})}. \tag{4}$$

And apply the pair-splitting strategy: If the s_{it} is the maximum ratio so far, save the current it into well-fitting iteration (k) and the ratio as the maximum ratio ($s_{\max}$); If $it = it_{\max}$ then go to Step 5; If $s_{it} = 1$, go to Step 5.

Step 4. Increment it by one and go back to Step 2.

Step 5. Calculate the completion time C_j of i-th job for both clusters according to following formulas:

a. Cluster I:

$$C_j = k \cdot p_{1j} + (k-1) \cdot p_{2j} + ... + p_k, \tag{5}$$

b. Cluster II:

$$C_j = l \cdot p_{1j} + (l-1) \cdot p_{2j} + ... + p_l. \tag{6}$$

Tabulate these values into two rows to get two pseudo machines (M_{p1}, M_{p2}).

Step 6. Apply Johnson's rule on two pseudo machines:

Apply Johnson's rule on two pseudo machines of n jobs to get the job sequence.

Step 7. Display the solution:

The $C_{\max}$ of particular job sequence from *Step 6* is the solution.

We coded CH, NEH, CDS, Palmer's slope index and Gupta's algorithms in PHP script. All PHP-coded algorithms have user-friendly interface with eventuality to select whether to run each heuristic itself or all together.

Experiments description and results

Computational experiments were performed on 120 instances with 10 each of one particular size using Taillard's benchmark problem datasets range from 20 to 500 jobs and 5 to 20 machines. The previous specified algorithms will develop an optional sequence using makespan as the performance criterion. Makespan is defined as the time required completing the set of jobs through all machines. The outputs of NEH algorithm were used as reference solutions for comparison purposes.

Performance measures

The performance results were interpreted by using a relative percent deviation (RPD) and average relative percent deviation (ARPD) for comparing the solutions of each algorithm to reference solutions. The average percentage relative deviation is given by:

$$\mathrm{ARPD} = \frac{1}{I} \cdot \sum_{i=1}^{i} \mathrm{RPD}_i, \tag{7}$$

$$\mathrm{RPD}_i = \frac{\mathrm{HS}_i - \mathrm{RS}_i}{\mathrm{RS}_i} \cdot 100\%, \tag{8}$$

where I – number of problem instances, HS_i – heuristic solution of problem instance i, RS_i – reference solution of problem instance i, RPD_i – percentage relative deviation of problem instance i.

Results analysis

The performance results of makespan for Taillard's 120 instances and different algorithms are at first presented in Table 1. The results show the computational values of ARPDs for each algorithm and for each problem dataset. The final line in Table 1 gives the overall average RPD values over all problem instances.

Table 1
Results of makespan expressed by APRDs.

Problem size	Sample size	CH ARPD	CDS ARPD	Gupta ARPD	Palmer ARPD
20 × 5	10	5.94	9.21	9.31	7.13
20 × 10	10	8.77	18.62	19.04	10.23
20 × 20	10	9.46	15.98	18.17	12.3
50 × 5	10	5.103	12.382	11.091	4.112
50 × 10	10	7.038	17.074	15.35	7.951
50 × 20	10	8.781	16.801	16.623	9.453
100 × 5	10	3.572	5.522	5.302	1.876
100 × 10	10	6.915	13.351	12.806	6.776
100 × 20	10	8.282	16.592	16.357	8.46
200 × 10	10	5.601	11.129	10.586	3.841
200 × 20	10	7.603	13.799	14.454	8.35
500 × 20	10	5.498	11.997	11.288	4.747
ΣARPD	120	6.88	13.54	13.36	7.1

The solutions of the four algorithms are compared with NEH optimal solutions for problems with size up to 20 machines and 500 jobs.

Secondly, in order to evaluate obtained result in further details our focus was to analyze ranges of relative percentage deviations from NEH optimal solutions. For this purpose, are commonly considered three statistical properties for each problem size such as minimal RPD value, maximum RPD value and APRD value. Then we can draw up the following 4 graphs (Figs 1–4) for each benchmarked algorithm.

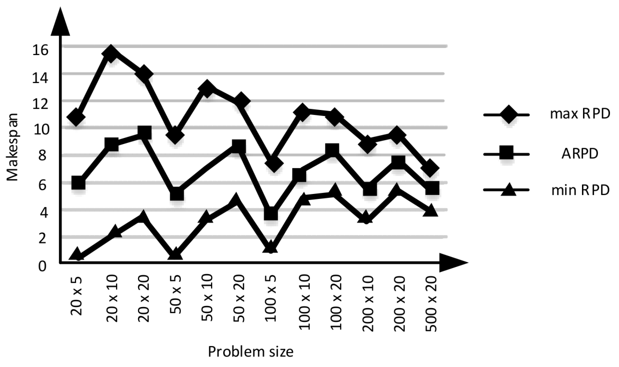

Fig. 1. Ranges of relative percentage deviations of CH from NEH optimal solutions.

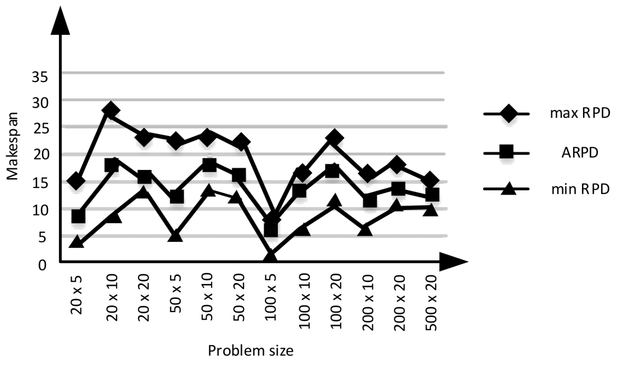

Fig. 2. Ranges of relative percentage deviations of CDS from NEH optimal solutions.

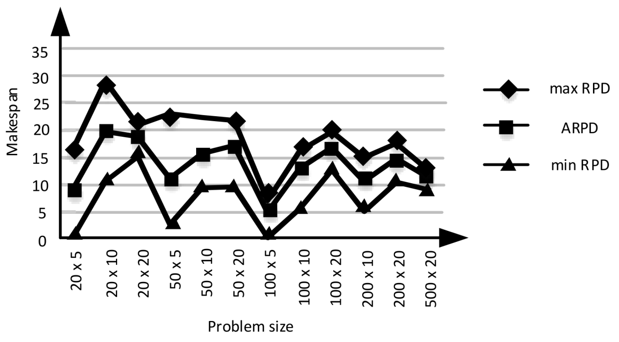

Fig. 3. Ranges of relative percentage deviations of GUPTA from NEH optimal solutions.

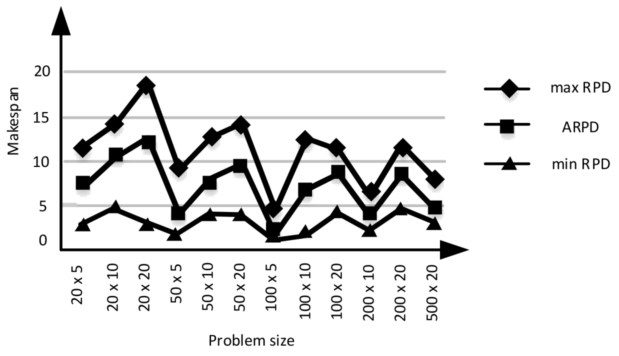

Fig. 4. Ranges of relative percentage deviations of PALMER from NEH optimal solutions.

The results indicate the following:

1. CH heuristic performed better than any of tested algorithms with the exception of NEH algorithm that was used as reference heuristic for this study. The second best makespan among the four tested algorithms was achieved by Palmer's slope index. Based on this finding, one could recommend for simpler scheduling problems in production management to apply the CH heuristics.

2. As it is evident from the Figs 1 to 4, the ranges of deviations from optimal makespan values tend to decrease with increasing problem size. Accordingly, it theoretically appears that this tendency is in line with general property of a statistical distribution that larger samples sizes have usually advantage of providing more data for researchers to work with, and are directly related to a statistic's margin of error. In a given case, the larger sizes of flow shop problems are better limiting the influence of outliers or extreme values as mean and can be considered to be more representative than smaller ones.

As practical consequence for each of tested algorithm, it can be observed that for bigger problem sizes, the cumulative relative deviations are smaller i.e. the performances become better and better. It might mean that benchmarked algorithms can be effectively applied in the production engineering tasks, especially, for scheduling and sequencing of manufacturing processes with bigger problem sizes.

Conclusions

In this paper, the heuristic approaches to solve sequencing problem with sequence-dependent jobs were presented and compared. Obtained results showed that compared algorithms are all capable of finding expected solutions of the given optimization criterion and the problem sizes. For given problem sizes, the CD algorithm generated better solutions than the well-known solutions in the literature. Several directions for future research deserve further exploration in relation to this frequent scheduling problem. For example, it would be useful to consider other performance criteria and investigate of multi-objective problems. Secondly, further study might be focused on larger problems. Thirdly, future studies can focus on the problems with realistic constraints such as sequence-dependent setup times and release times.

This paper has been supported by VEGA project No. 1/0419/16 granted by the Ministry of Education of the Slovak Republic.

References

[1] Ruiz R., Stützle T., *A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem*, European Journal of Operational Research, 177(3), 2033–2049, 2007.

[2] Agarwal A., Colak S., Eryarsoy E., *Improvement heuristic for the flow-shop scheduling problem: An adaptive-learning approach*, European Journal of Operational Research, 169(3), 801–815, 2006.

[3] Johnson S.M., *Optimal two- and three-stage production schedules with setup times included*, Nav Res Log Q 1, pp. 61–68, 1954.

[4] Campbell H.G., Dudek R.A., Smith M.L., *A heuristic algorithm for the n-job,m-machine sequencing problem*, Management Science, 16, 10, 630–637, 1970.

[5] Koulamas C., *A new constructive heuristic for the flowshop scheduling problem European*, Journal of Operational Research Society, 105, 66–71, 1998.

[6] Yinga K.C., Liao C.J., *An ant colony system for permutation flow-shop sequencing*, Computing Operation Research, 31(5), 791–801, 2004.

[7] Taillard E, *Benchmarks for basic scheduling problems*, European Journal of Operation Research, 64, 278–285, 1993.

[8] Lin S.W., Ying K.C., *Optimization of makespan for no-wait flowshop scheduling problems using efficient matheuristics*, Omega, 64, 115–125, 2015.

[9] Nowicki E., Smutnicki C., *A fast tabu search algorithm for the permutation flow-shop problem*, European Journal of Operational Research, 91(1), 160–175, 1996.

[10] Modrak V., Pandian R.S., *Flow shop scheduling algorithm to minimize completion time for n-jobs m-machines problem*, Tehnicki Vjesnik, 17(3), 273–278, 2010.

[11] Boukef H., Benrejep M., Borne P., *A proposed genetic algorithm coding forflow-shop scheduling problems*, International Journal Computing Communication Control, 2(3), 229–240, 1953.

[12] Dima I.C., Gabrara J., Modrak V., Piotr P., Popescu C., *Using the expert systems in the operational management of production*, 11th WSEAS International Conference on Mathematics and Computers in Business and Economics, pp. 307–312, 2010.

[13] Chandra P., Mehta P., Tirupati D., *Permutation flow shop scheduling withearliness and tardiness penalties*, International Journal of Production Research, 47(20), 5591–5610, 2009.

[14] Semanco P., Modrak V. *A comparison of constructive heuristics with the objective of minimizing makespan in the flow-shop scheduling problem*, Acta Polytechnica Hungarica, 9(5), 177–190, 2012.

[15] Papadimitriou C.H., Kanellakis P.C., *Flowshop scheduling with limited temporary storage*, Journal of the ACM (JACM), 27(3), 533–549, 1980.

[16] Modrak V., Mandulak J., *Mapping Development of MES Functionalities*, ICINCO-SPSMC – 6th International Conference on Informatics in Control, Automation and Robotics, pp. 244–247, 2009.

[17] Yenisey M.M., Yagmahan B., *Multi-objective permutation flow shop scheduling problem: Literature review, classification and current trends*, Omega, 45, 119–135, 2014.

[18] Arroyo J.C., Souza Pereira A.A., *A GRASP heuristic for the multi-objective permutation flowshop scheduling problem*, International Journal of Advanced Manufacturing Technology, 55, 741–753, 2011.

[19] Framinan J.M., Leisten R., *A multi-objective iterated greedy search for flowshop scheduling with makespan and flowtime criteria*, OR Spectrum, 30, 787–804, 2008.

[20] Allahverdi A., *A new heuristic for m-machine flowshop scheduling problem with bicriteria of makespan and maximum tardiness*, Computers & Operations Research, 31(2), 157–180, 2004.

[21] Chou F.D., Lee C.E., *Two-machine flowshop scheduling with bicriteria problem*, Computers & Industrial Engineering, 36, 549–564, 1999.

[22] Gupta J.N.D., Hennig K., Werner F., *Local search heuristics for two-stage flow shop problems with secondary criterion*, Computers & Operations Research, 29(2), 123–149, 2002.

[23] T'kindt V., Gupta J.N.D., Billaut J.C., *Two-machine flowshop scheduling with a secondary criterion*, Computers & Operations Research, 30(4), 505–526, 2003.

[24] Arroyo J.C., Armentano V.A., *A partial enumeration heuristic for multi-objective flowshop scheduling problems*, Journal of the Operational Research Society, 55, 1000–1007, 2004.

[25] Minella G., Ruiz R., Ciavotta M., *Restarted Iterated Pareto Greedy algorithm for multi-objective flowshop scheduling problems*, Computers & Operations Research, 38, 1521–1533, 2011.

[26] Pasupathy T., Rajendran C., Suresh R.K., *A multi-objective genetic algorithm for scheduling in flow shops to minimize the makespan and total flow time of jobs*, International Journal of Advanced Manufacturing Technology, 27, 804–815, 2006.

[27] Rahimi-Vahed A.R., Mirghorbani S.M., *A multi-objective particle swarm for a flowshop scheduling problem*, Journal of Combinatorial Optimization, 13(1), 79–102, 2007.

[28] Allouche M.A., Aouni B., Martel J.M., Loukil T., Rebaż A., *Solving multi-criteria scheduling flow shop problem through compromise programming and satisfaction functions*, European Journal of Operational Research, 192, 460–467, 2009.

[29] Tran T.H., Ng K.M., *A water-flow algorithm for flexible flow shop scheduling with intermediate buffers*, Journal Scheduling, 4(5), 483–500, 2010.

[30] Godard D., Laborie P., Nuitjen W., *Randomized large neighborhood search for cumulative scheduling*, Proceedings of ICAPS-05, pp. 81–89, 2005.

[31] Dupont de Dinechin B., *Time-indexed formulations and a large neighborhood search for the resource-constrained modulo scheduling problem*, Baptiste P., Kendall G., Munier-Kordon A., Sourd F. [Eds], 3rd Multidisciplinary International Scheduling conference: Theory and Applications (MISTA), 2007.

[32] Lu C., Gao L., Li X., Pan Q., Wang Q., *Energy-efficient permutation flow shop scheduling problem using a hybrid multi-objective backtracking search algorithm*, Journal of Cleaner Production, 144, 228–238, 2017.

[33] Bai D., Liang J., Liu B., Tang M., Zhang Z.H., *Permutation flow shop scheduling problem to minimize nonlinear objective function with release dates*, Computers & Industrial Engineering, 112, 336–347, 2017.

[34] Rajendran S., Rajendran C., Leisten R., *Heuristic rules for tie-breaking in the implementation of the NEH heuristic for permutation flow-shop scheduling*, International Journal of Operational Research, 28(1), 87–97, 2017.

[35] Deng J., Wang L., *A competitive memetic algorithm for multi-objective distributed permutation flow shop*

scheduling problem, Swarm and Evolutionary Computation, 32, 121–131, 2017.

[36] Li X., Ma S., *Multiobjective Discrete Artificial Bee Colony Algorithm for Multiobjective Permutation Flow Shop Scheduling Problem With Sequence Dependent Setup Times*, IEEE Transactions on Engineering Management, 64(2), 149–165, 2017.

[37] Qian B., Li Z.C., Hu R., *A copula-based hybrid estimation of distribution algorithm for m-machine reentrant permutation flow-shop scheduling problem*, Applied Soft Computing, 61, 921–934, 2017.

[38] Rossit D.A., Tohmé F., Frutos M., *The non-permutation flow-shop scheduling problem: a literature review*, Omega, 2017.

[39] Grabowski J., Pempera J., *Sequencing of jobs in some production system*, European Journal of Operation Research, 125(3), 535–550, 2000.