Management and Production Engineering Review
Volume 5 • Number 4 • December 2014 • pp. 26–35
DOI: 10.2478/mper-2014-0033

ROUTING FLOW-SHOP WITH BUFFERS AND READY TIMES – COMPARISON OF SELECTED SOLUTION ALGORITHMS

Jerzy Józefczyk[1], Michał Markowski[1], Ljazat Balgabaeva[2]

[1] *Wroclaw University of Technology, Institute of Informatics, Wroclaw, Poland*
[2] *Kazakh Technical University, Department of Information Processing, Almaty, Kazakhstan*

Corresponding author:
Jerzy Józefczyk
Wroclaw University of Technology
Institute of Informatics
Wybrzeże Wyspiańskiego 27, 50-370 Wroclaw, Poland
phone: (+48) 71 320-29-79
e-mail: Jerzy.Jozefczyk@pwr.edu.pl

Received: 5 September 2014
Accepted: 12 October 2014

Abstract
This article extends the former results concerning the routing flow-shop problem to minimize the makespan on the case with buffers, non-zero ready times and different speeds of machines. The corresponding combinatorial optimization problem is formulated. The exact as well as four heuristic solution algorithms are presented. The branch and bound approach is applied for the former one. The heuristic algorithms employ known constructive idea proposed for the former version of the problem as well as the Tabu Search metaheuristics. Moreover, the improvement procedure is proposed to enhance the quality of both heuristic algorithms. The conducted simulation experiments allow evaluating all algorithms. Firstly, the heuristic algorithms are compared with the exact one for small instances of the problem in terms of the criterion and execution times. Then, for larger instances, the heuristic algorithms are mutually compared. The case study regarding the maintenance of software products, given in the final part of the paper, illustrates the possibility to apply the results for real-world manufacturing systems.

Keywords
manufacturing systems, operations research, complex systems, optimization problems, scheduling algorithms, routing algorithms, heuristics, computer simulation.

Introduction

The particular problem considered in the paper deals with foundations of manufacturing systems management. The main activity of every manufacturing system is accompanied by interconnected auxiliary activities like storage and transportation which have to be taken into account while managing such systems as a whole. According to the most popular and earliest approach, every activity is managed separately. However, it is obvious that the joint derivation of management decisions for component activities can improve the action of manufacturing systems in terms of the maximization of profit or the minimization of cost (execution time). Corresponding branches of operations research are foundations for the development of management methods and algorithms useful for real-world manufacturing systems.

The idea of integration and joint consideration of different activities connected with manufacturing has been developed since several last years in the framework of operations research, in general and combinatorial optimization, in particular. Production, manufacturing, logistic and service systems are mainly pointed out as prospective areas of applications. Location, vehicle routing, task scheduling, queueing, assignment, inventory, resource allocation are the most important combinatorial optimization problems which management algorithms are suitable for manufacturing systems. Many combinations of these problems are investigated and reported in the literature. Let us mention some of them as the example: location routing problem [1, 2], lo-

cation scheduling problem [3, 4], inventory location problem [5], inventory routing problem [6, 7], routing scheduling problem [8–11], production inventory problem [12], production transportation problem [13]. The derivation of useful solution algorithms for real-world applications takes advantage of the progress in solution methods and techniques in operations research and first of all in computer technology.

The particular routing scheduling problem is considered in the paper being the combination of known travelling salesman (TSP) and flow-shop subproblems. Both component sub-problems together with their special cases are classical, thoroughly studied topics in operations research. It is worth noting the flow-shop with setup times and with batches which are similar to the version considered in the paper [14–18]. It is assumed in the routing flow-shop that machines being executors of jobs are not stationary, but they are able to move among jobs. The flow of jobs is sometimes impossible due to the difficulties in handling or relocating jobs that are too big or too small, too heavy or cannot be moved due to technological limitations. Such situations may occur for the production of ships, big wagons, cars or small parts like transistors. In these cases, moving machines can be used to drive from one job located at its workstation to another one. For example, to build a ship, four machines can work alongside the ship: the first machine polishes the surface of the ship for further processing, the second one rivets metal plates, the third one paints with the anticorrosive paint, and the last one paints the ship with the final color. Parts of the ship cannot be moved due to its size; however, mobile machines can move from one area to another one. Taking into account the possibility of machine movement while considering task (job) scheduling leads to so called routing scheduling problem or task scheduling with moving machines (executors), e.g. [10]. The version with moving machines can concern every particular task scheduling problem. Some results for parallel machines can be found in [8, 9, 19].

This work is focused on the flow-shop problem and extends results presented in [10] where the simple approximation algorithm is proposed together with the evaluation of its quality for the routing flow-shop problem with unlimited buffers, without ready times and with equal driving speeds of machines. We propose to use this algorithm to solve the considered problem, i.e. the version with non-zero ready times and different speeds of machines.

The routing flow-shop problem without buffers is considered in [20]. The authors present a recurrent procedure for the calculation of makespan, which is used by the heuristic greedy algorithm. The results of this algorithm are compared for small instances of the problem to the optimal solutions generated via simple enumeration.

A 10/7-approximation algorithm for two-machine routing flow-shop is given in [21]. The same work contains also another approximation algorithm for m-machine routing open-shop as well as for the routing flow-shop with unlimited buffers and without ready times. Both algorithms deal with the flow-shop better than those presented in [10, 22]. In [21] the NP-hardness of two-machine routing flow-shop is proved via the reduction from the partitioning problem.

The uncertain version of classical task scheduling with routing, when the execution times are not precise, but the corresponding intervals of given bounds are only known, is investigated in [23, 24]. The objective function based on the regret is used. The Tabu Search (TS) and Simulated Annealing solution algorithms are developed and compared.

Investigations of this work refer to the version of the flow-shop with non-zero ready (release) times, see e.g. [25], where the branch and bound algorithm is proposed to solve three-machine problem with makespan as the criterion.

The flow-shop problems with routing are more complex than their classical versions because driving times and sometimes driving limitations have to be taken into account. It is important to note that the flow-shop with routing can be considered as the difficult and very rare investigated version of so called task scheduling with setup times and sequence dependent setups, e.g. [14].

The reminder of this work is organized as follows. Three solution algorithms are presented after the formulation of the considered routing flow-shop as the combinatorial optimization problem. The first, exact algorithm is based on the branch and bound approach (B&B). Two remaining heuristic algorithms are developed on different bases. The first one directly uses the idea proposed in [10], i.e. it applies the solution algorithm of the multiple TSP. The second one employs the TS metaheuristics. The improvement procedure is proposed for both heuristic algorithms, which can foster the main algorithms by the results of partial sub-problems of smaller sizes solved by the branch and bound algorithm. The next section presents the evaluation of all algorithms via simulation experiments. The conclusions following the presentation of the case study concerning the maintenance of software products complete the paper.

Routing flow-shop problem

Let us consider the flow-shop problem with m machines and n jobs where $\mathbf{M} = \{M_1, M_2, \ldots, M_m\}$ and $\mathbf{J} = \{J_1, J_2, \ldots, J_n\}$ are sets of machines and jobs, respectively. Indices i, j denote the current machine, job, correspondingly. A workstation is defined as the place where job is located. There is no particular difference between the job and the workstation, however the notion 'workstation' refers to the localization of job. A depot as the workstation where all machines start and finish their work, and no activity is performed, is denoted by J_{n+1}. All jobs and the depot constitute a set $\overline{\mathbf{J}} = \{J_1, \ldots, J_n, J_{n+1}\}$. Every job is composed of m operations being its parts and performed by consecutive machines. Operation $O_{i,j}$ refers to the part of job J_j, which is performed by machine M_i. Operations within particular job J_j are performed by machines in the fixed order and constitute a sequence $(O_{1,j}, O_{2,j}, \ldots, O_{m,j})$. The order of machines denoted as $(M_1, M_2, \ldots, M_m)$ is given unlike the order of jobs undergoing the decision. Due to the movement of machines, every operation is composed of two parts: driving of a machine between the workstations and performing of an activity at the workstation. We denote by $\widehat{p}_{i,l,j}$ and $p_{i,j}$ the driving-up time of machine M_i from workstation J_l to workstation J_j, and the execution time of activity $O_{i,j}$, respectively. In a consequence, $\widetilde{p}_{i,l,j} = p_{i,j} + \widehat{p}_{i,l,j}$ is the execution time of operation $O_{i,j}$. The ready time for job J_j denoted as r_j means that the job cannot start before this time elapses.

In order to formulate the corresponding optimization problem, the decision variable being a sequence (permutation) of machines' routes is defined as $\Pi = (J_{\pi_o}, J_{\pi_1}, \ldots, J_{\pi_{n+1}}) \in \mathbf{\Pi}$ where $\mathbf{\Pi}$ is the set of all feasible permutations. Moreover, $(\pi_1, \pi_2, ..., \pi_n)$ is the permutation of $(1,\ 2,\ ...,\ n)$, $\pi_0 = \pi_{n+1} = n+1$ represent the depot, $\pi_i \neq \pi_k$, and $\pi_i = j$ means that job J_j is performed as the i-th. This work refers to the permutation version of the flow-shop problem both in its classical version and with routing, e.g. [26, 27], so we assume that every machine follows the same sequence and performs jobs in the same order.

Two cases of the routing flow-shop can be considered with respect to buffers as the equipment of workstations [20]. Workstations without buffers can only host one machine, i.e. the machine that performs an activity. No additional machines are allowed to wait or stop at the workstation where an activity is currently performed by another machine. Such a constraint influences the calculation of the makespan. Before it drives up to the next workstation, the machine has to wait for leaving this workstation by the previous machine. This requirement does not exist in the case with buffers, which is discussed in the work. Additionally, it is assumed that buffers have unlimited capacity. The example of the routing flow-shop for $n = 4$, $m = 3$ and $\Pi = (J_{\pi_0}, J_{\pi_1}, J_{\pi_2}, J_{\pi_3}, J_{\pi_4}, J_{\pi_5})$, $(\pi_0, \pi_1, \pi_2, \pi_3, \pi_4, \pi_5) = (5,\ 3,\ 4,\ 2,\ 1,\ 5)$ is presented in Fig. 1.

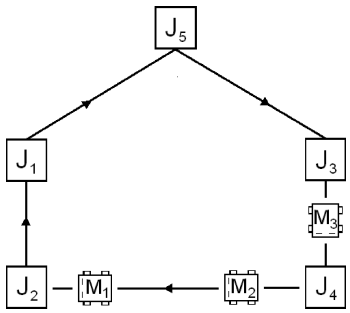

Fig. 1. Example of a layout of workstations and machines' routes.

Problem formulation

The makespan, being the time moment when the last job is completed, serves as the criterion evaluating the decision variable Π. We propose to calculate the makespan denoted as $C_{\max}(\Pi)$ recurrently. Let us denote by $C(\Pi, i, k)$ the time moment when machine M_i can start to move to the next workstation $J_{\pi_k} \in \overline{\mathbf{J}}$ where index k refers to the position in sequence Π. This time moment can be calculated for machines M_i, $i = 2,\ 3,\ ...,\ m$ as:

$$C(\Pi, i, k) = \max[C(\Pi, i, k-1) + \widetilde{p}_{i,\pi_{k-2},\pi_{k-1}};\ C(\Pi, i-1, k) + \widetilde{p}_{i-1,\pi_{k-1},\pi_k} - \widehat{p}_{i,\pi_{k-1},\pi_k}] \quad (1)$$

for $k = 2,\ 3,\ ...,\ n$, and as:

$$C(\Pi, i, 1) = \max[C(\Pi, i-1, 1) + \widetilde{p}_{i-1,\pi_0,\pi_1} - \widehat{p}_{i,\pi_0,\pi_1}; 0] \quad (2)$$

for $k = 1$.

The start times of jobs on machine M_1 are calculated differently due to the ready times u_h:

$$C(\Pi, 1, k) = \max[C(\Pi, 1, k-1) + \widetilde{p}_{1,\pi_{k-2},\pi_{k-1}}; r_{\pi_k} - \widehat{p}_{1,\pi_{k-1},\pi_k}] \quad (3)$$

for $k = 2,\ 3,\ \ldots, n+2$, $r_{n+1} = r_{n+2} = 0$ and

$$C(\Pi, 1, 1) = \max[0; r_{\pi_1} - \widehat{p}_{1,\pi_0,\pi_1}]. \quad (4)$$

Finally, the makespan is the maximum of returns to the depot by all machines

$$C_{\max}(\Pi) = \max_{i=1,2,\ldots,m}[C(\Pi, i, n) + \widetilde{p}_{i,\pi_{n-1},\pi_n} + \widehat{p}_{i,\pi_n,\pi_{n+1}}]. \quad (5)$$

So, the considered routing flow-shop problem consists in the determination of such sequence Π to

minimize $C_{\max}(\Pi)$ for given: $\mathbf{M}, \overline{\mathbf{J}}$, $p_{i,j}$, $\widehat{p}_{i,l,j}$, r_j, $i = 1, 2, ..., m$, $l, j = 1, 2, ..., n$.

As the result, Π^* and $C_{\max}(\Pi^*)$ are obtained. This optimization problem is at least NP-hard due to the NP-hardness of it classical counterpart, i.e. the version without routing [28]. The paper is focused on efficient heuristic solution algorithms which are presented in the subsequent section.

Exact and heuristic solution algorithms

Three algorithms are presented in this section: the exact algorithm, referred to as BB, based on the branch and bound approach and two heuristic algorithms. The first heuristic algorithm, referred to as Alg1, was proposed in [10] for the simpler version of the problem assuming the same ready times and the same velocities of machines. As it is shown in [29], this algorithm can be easily adopted for the considered problem but without the approximation property reported in [10]. The second heuristic algorithm uses TS metaheuristics. Moreover, the improvement procedure with the usage of the branch and bound approach is proposed. After joining it with Alg1 and Alg2, two new hybrid algorithms are obtained called AlgI1 and AlgI2, respectively.

Branch and bound exact algorithm

The branching procedure enables us checking all feasible permutations Π, and it is illustrated as moves along a tree composed of vertices denoting partial permutations. The leaves of the tree stand for full permutations. The lower bounds of makespan $C_{\max}$ referred to as C_{LB}, are calculated at every vertex, and they make possible to limit the searching procedure along the tree. Let us denote by $\Pi(v)$ a partial solution (permutation) ready at vertex v, which contains jobs belonging to the set $\mathbf{J}(v) \subset \mathbf{J}$. The jobs not scheduled yet form the set $\mathbf{J}'(v) = \mathbf{J} \backslash \mathbf{J}(v)$. Then, jobs from sets $\mathbf{J}(v)$ and $\mathbf{J}'(v)$ belong to sub-solution $\Pi(v) = (J_{n+1}, J_{\pi_1}, ..., J_{\pi_i}, ..., J_{\pi_{|\mathbf{J}(v)|}})$, $J_{\pi_i} \in \mathbf{J}(v)$ and $\Pi'(v) = \Pi \backslash \Pi(v)$, respectively.

Then, the lower bounds are calculated according to the formula:

$$\begin{aligned} C_{\mathrm{LB}}(\Pi, k) = & \max_{i=1,2,...,m} [C(\Pi, i, k+1)] \\ & + \sum_{J_{\pi_l} \in \mathbf{J}'(v)} p_{m,\pi_l} + \sum_{J_{\pi_k} \in \mathbf{J}'(v)} p^k_{\min} - p_{\max} \\ & + \min_{J_{\pi_l} \in \mathbf{J}'(v)} \widetilde{p}_{m,\pi_l,n+1} + \max[0; \\ & \min_{J_{\pi_l} \in \mathbf{J}'(v)} r_{\pi_l} - \max_{i=1,2,...,m} (C(\Pi, i, k+1)) \\ & - \max_{J_{\pi_l} \in \mathbf{J}'(v)} \widetilde{p}_{m,\pi_{|J(v)|},\pi_l}], \end{aligned} \tag{6}$$

and

$$p^k_{\min} = \min_{J_{\pi_l} \in \mathbf{J}'(v)} \tilde{p}_{m,\pi_l,\pi_k}, \quad p_{\max} = \max_{J_{\pi_k} \in \mathbf{J}'(v)} p^k_{\min}.$$

The right hand side of (6) includes respectively: the makespan of partial solution $\Pi(v)$, the sum of execution times of activities related to the remaining set of jobs $\mathbf{J}'(v)$, the sum of minimum driving-up times to every remaining job from $\mathbf{J}'(v)$ adjusted by $p_{\max}$, the minimum of driving-up times from locations of jobs belonging to $\mathbf{J}'(v)$ to the depot; the minimum ready time less the makespan of partial solution and less the longest driving-up time from the last job of partial solution to any remaining job.

TSP-based heuristic algorithm

The algorithm Alg1 is based on the solution of TSP when the machines and workstations are treated as salesmen and visited cities, respectively. Let us denote by $\Pi_\varepsilon = (J_{\pi_o}, J_{\pi_1}, \ldots, J_{\pi_{n+1}})$ ε – approximate solution of TSP, being the sequence of all workstations with the beginning and the end at the depot (all machines work according to the same sequence Π_ε), $\Pi^{\mathrm{T}}_\varepsilon = (J_{\pi_{n+1}}, J_{\pi_n}, \ldots, J_{\pi_0})$ – the sequence of workstations reverse to π_ε, $\overline{\Pi}$ – result of Alg1. Then, Alg1 is composed of two steps.

Algorithm Alg1

Input: algorithm for TSP returning the solution Π_ε

Output: $\overline{\Pi}$, $C_{\max}(\overline{\Pi})$

1. Determine Π_ε and $\Pi^{\mathrm{T}}_\varepsilon$, as well as calculate $C_{\max}(\Pi_\varepsilon)$, and $C_{\max}(\Pi^{\mathrm{T}}_\varepsilon)$.
2. If $C_{\max}(\Pi_\varepsilon) \leq C_{\max}(\Pi^{\mathrm{T}}_\varepsilon)$ set $\overline{\Pi} = \Pi_\varepsilon$, otherwise set $\overline{\Pi} = \Pi^{\mathrm{T}}_\varepsilon$, and calculate $C_{\max}(\overline{\Pi})$.

Any known ε – approximation algorithm solving TSP can be used to obtain Π_ε. The computational complexity of Alg1 is determined by the used ε – approximation algorithm.

Tabu Search algorithm

Let us start with introducing additional notions and notation. We assume that moves as crucial elements of TS are limited only to insertions. The move $v_{\pi_i,\pi_k}(\Pi) \in \mathbf{\Pi}$ denotes the replacement of positions between jobs π_i and π_k in solution Π. The new solution $v_{\pi_i,\pi_k}(\Pi)$ is obtained as the result, which can denote also another solution generated after performing the move. The moves on Π constitute the $\overline{n}$ – element set $\mathbf{V}(\Pi)$ called also the neighborhood of Π. The tabu list $TL = (T_1, ..., T_l, ..., T_L)$ of L elements T_l contains attributes of solutions and moves referred to as $a_v(\Pi, v) \in \mathbf{A}_v(\Pi, v)$ where $\mathbf{A}_v(\Pi, v)$ is the set of attributes defined in the paper as $A(\Pi, v) = \{(\pi_{i-1}, \pi_i), (\pi_i, \pi_{i+1}), i = 1, 2, ..., n\}$. All moves for Π are divided into forbidden $\mathbf{V}_2(\Pi) \subseteq \mathbf{V}(\Pi)$ and

free $\mathbf{V}_1(\Pi) \subseteq \mathbf{V}(\Pi)$ ones, i.e. $\mathbf{V}_1(\Pi) \cup \mathbf{V}_2(\Pi) = \mathbf{V}(\Pi)$. The move for which $C_{\max}(v_{\pi_i,\pi_k}(\Pi)) \leq C_{\max}(\widetilde{\Pi})$ holds, where $C_{\max}(\widetilde{\Pi})$ is the makespan for the current best solution $\widetilde{\Pi}$ generated by Alg2, is called the prospective forbidden move and is an element of the subset $\mathbf{V}_{21}(\Pi) \subseteq V_2(\Pi)$ of prospective forbidden moves. Other elements of $\mathbf{V}_2(\Pi)$ form a subset $\mathbf{V}_{22}(\Pi) = \mathbf{V}_2(\Pi) \backslash \mathbf{V}_{21}(\Pi)$ which comprises non-prospective forbidden moves. Alg2 starts with initial population Π_0 generated randomly or being the result of another heuristic algorithm and ends after N iterations.

Algorithm Alg2

Input: $TL = \varnothing$, $\kappa = 1$, $\widetilde{\Pi} = \Pi = \Pi_0$, $\overline{n}$, N

Output: $\widetilde{\Pi}$, $C_{\max}(\widetilde{\Pi})$

1. Determine $V(\Pi)$ performing $\overline{n}$ times the moves $v_{\pi_i,\pi_k}(\Pi)$, where π_i, π_k are randomly generated indices of jobs.

2. Determine sets $V_1(\Pi)$, $V_{21}(\Pi)$, $V_{22}(\Pi)$, and go to Step 4 if $V_1(\Pi) \cup V_{21}(\Pi) = \varnothing$.

3. Find the move $v^*_{\pi_i,\pi_k}(\Pi)$ minimizing $C_{\max}(\Pi)$ i.e. $C_{\max}(v^*_{\pi_i,\pi_k}(\Pi)) = \min_{v_{\pi_i,\pi_k}(\Pi) \in V_1(\Pi) \cup V_{21}(\Pi)} C_{\max}(\Pi)$ and go to Step 5.

4. If $|V_{22}(\Pi)| = 1$ denote element of $V_{22}(\Pi)$ as $v^*_{\pi_i,\pi_k}(\Pi)$, and go to Step 5 else repeat Steps 4a and 4b until $V_1(\Pi) \cup V_{21}(\Pi) \neq \varnothing$ or $T_l = T_L$, $l = 1, 2, ..., L$:

a. Set $TL = TL \oplus T_L$, i.e. join T_L to the end of TL.

b. Determine new sets $V_1(\Pi)$, $V_{21}(\Pi)$, $V_{22}(\Pi)$ and go to Step 3.

5. Set $TL = TL \oplus a_v(\Pi, v^*_{\pi_i,\pi_k}(\Pi))$, $\Pi = v^*_{\pi_i,\pi_k}(\Pi)$.

6. Calculate $C_{\max}(\Pi)$.

If $C_{\max}(\Pi) < C_{\max}(\widetilde{\Pi})$ set: $\widetilde{\Pi} = \Pi$, $C_{\max}(\widetilde{\Pi}) = C_{\max}(\Pi)$, and $\kappa = 1$.

7. If $\kappa < N$ set $\kappa = \kappa + 1$, and go to Step 1 else stop the algorithm.

Improvement procedure

This procedure consists in the application of B&B to the solution (permutation) attained by any heuristic algorithm. Let us present it for $\widetilde{\Pi}$ obtained by Alg2 (the case of $\overline{\Pi}$ produced by Alg1 follows accordingly). More precisely, B&B is simultaneously used for sub-permutations $\widetilde{\Pi}_\lambda$ of $\widetilde{\Pi}$ where $\widetilde{\Pi} = \left(\widetilde{\Pi}_1, ..., \widetilde{\Pi}_\lambda, ..., \widetilde{\Pi}_\Lambda\right)$, and $\Lambda = \lceil n/\eta \rceil$ is the number of sub-permutations of the length η. Every sub-permutation consists of a part of $\widetilde{\Pi}$, i.e.

$$\widetilde{\Pi}_\lambda = (J_0, J_{\pi_{(\lambda-1)\eta+1}}, J_{\pi_{(\lambda-1)\eta+2}}, .., J_{\pi_{\lambda\eta}}, J_{n+1}),$$
$$\lambda = 1,\ 2, .,\ \Lambda - 1,$$
$$\widetilde{\Pi}_\Lambda = (J_0, J_{\pi_{(\Lambda-1)\eta+1}}, J_{\pi_{(\Lambda-1)\eta+2}}, ..., J_{\pi_n}, J_{n+1}).$$

Then, after excluding the excessive depots necessary for partial solutions, all partial solutions are merged into one final solution.

B&B based improvement procedure (IP)

Input: $\widetilde{\Pi}$, η

Output: $\widetilde{\Pi}_{BB}$, $C_{\max}(\widetilde{\Pi}_{BB})$

1. Divide permutation $\widetilde{\Pi}$ into Λ sub-permutations $\widetilde{\Pi}_1$, $\widetilde{\Pi}_2$, ..., $\widetilde{\Pi}_\lambda$, ..., $\widetilde{\Pi}_\Lambda$.

2. Repeat for $\lambda = 1,\ 2, ..., \Lambda$:

Solve the routing scheduling problem using the BB exact algorithm for the following data: $\mathbf{M}, \mathbf{J}_\lambda = (J_{\pi_i})_{i \in \overline{(\lambda-1)\eta+1, \lambda\eta}}$, $\lambda = 1,\ 2, ..., \Lambda - 1$, $\mathbf{J}_\Lambda = (J_{\pi_i})_{i \in \overline{(\Lambda-1)\eta+1, n}}$, and corresponding values of $p_{i,j}$, $\widehat{p}_{i,l,j}$, r_j, to obtain

$$\widetilde{\Pi}_{\lambda,\mathrm{BB}} = (J_0, J_{\widetilde{\pi}_{(\lambda-1)\eta+1,\mathrm{BB}}}, J_{\widetilde{\pi}_{(\lambda-1)\eta+2,\mathrm{BB}}}, ...,$$
$$J_{\widetilde{\pi}_{\lambda\eta,\mathrm{BB}}}, J_{n+1}), \qquad \lambda = 1,\ 2, .., \Lambda - 1,$$

and

$$\widetilde{\Pi}_{\Lambda,\mathrm{BB}} = (J_0, J_{\widetilde{\pi}_{(\Lambda-1)\eta+1,\mathrm{BB}}}, J_{\widetilde{\pi}_{(\Lambda-1)\eta+2,\mathrm{BB}}}, ...,$$
$$J_{\widetilde{\pi}_{n,\mathrm{BB}}}, J_{n+1}).$$

3. Compose the final solution (after excluding excessive depots necessary for partial solutions) as $\widetilde{\Pi}_{\mathrm{BB}} = (J_0, \widetilde{\Pi}_{1,\mathrm{BB}}, \widetilde{\Pi}_{2,\mathrm{BB}}, ..., \widetilde{\Pi}_{\Lambda,\mathrm{BB}}, J_{n+1})$, and calculate $C_{\max}(\widetilde{\Pi}_{\mathrm{BB}})$.

Hybrid algorithms

When launching the improvement procedure with $\overline{\Pi}$, being the result of Alg1, as the initial solution, we have hybrid algorithm AlgI1. Analogously, merging of Alg2 with the improvement procedure gives AlgI2 as the result.

Evaluation of solution algorithms

All presented algorithms were coded in C# and evaluated during simulation experiments. The computations were performed on Intel Core U7300 1.3 GHz and 4.00 GB of RAM. The presented results are mean values of ten independent runs of the algorithms for different randomly generated data sets. Each data set was randomly generated according to the rectangular distributions from the intervals: $p_{i,j}, \widehat{p}_{i,l,j} \in \{1,\ 2,\ ...,\ 50\}$, $r_j \in \{0, 1,\ ...,\ 20\}$. The algorithms were launched for the following parameters: Alg1 – the cheapest insertion method as the algorithm solving TSP, [30]; Alg2 – $\overline{n} = 6$, $N = 6000$ and Π_0 generated randomly; IP – $\eta = 8$. Due to the lack of basis for comparison in the form of benchmarks or results of other algorithms, the heuristic algorithms were mutually compared for greater in-

stances and were referred to the BB exact algorithm for small instances. The values of criterion $C_{\max}$ and the execution times T are the basis for evaluation. The notation is proposed to distinguish the algorithms: $C_{\max,x}$, T_x, $x \in \{\text{BB, Alg1, AlgI1, Alg2, AlgI2}\}$. The results for different numbers of jobs n and machines m are presented in Tables 1–4 where profits $\delta_{C,1}$, $\delta_{C,2}$ in terms of $C_{\max}$ caused by the application of IP in Alg1 and Alg2 are also inserted, where $\delta_{C,1} = \frac{C_{\max,\text{Alg1}} - C_{\max,\text{AlgI1}}}{C_{\max,\text{AlgI1}}} 100\%$ and $\delta_{C,2} = \frac{C_{\max,\text{Alg2}} - C_{\max,\text{AlgI2}}}{C_{\max,\text{AlgI2}}} 100\%$.

Table 1
Comparison of heuristic and exact algorithms.

m	n	$C_{\max,x}$					$\delta_{C,1}$	$\delta_{C,2}$	T_x				
		BB	Alg1	AlgI1	Alg2	AlgI2			BB	Alg1	AlgI1	Alg2	AlgI2
2	2	105	105	105	105	105	0%	0%	< 0.01	< 0.5	< 0.5	< 0.5	< 0.5
2	3	150	150	150	150	150	0%	0%	< 0.01	< 0.5	< 0.5	< 0.5	< 0.5
2	4	178	185	178	178	178	4%	0%	< 0.01	< 0.5	< 0.5	< 0.5	< 0.5
2	5	206	209	206	207	206	1%	0%	< 0.01	< 0.5	< 0.5	1	1
2	6	239	247	239	249	239	3%	4%	< 0.01	< 0.5	< 0.5	1	1
2	7	274	282	274	282	274	3%	3%	< 0.01	< 0.5	< 0.5	2	2
2	8	306	318	306	319	306	4%	4%	< 0.01	< 0.5	< 0.5	2	3
2	9	336	349	336	344	336	4%	2%	< 0.01	< 0.5	< 0.5	3	3
2	10	369	382	374	380	374	2%	2%	2	< 0.5	< 0.5	4	4
2	11	400	413	406	419	410	2%	2%	18	< 0.5	< 0.5	4	5
2	12	421	433	426	437	430	2%	2%	53	< 0.5	< 0.5	5	5
2	13	461	478	470	479	470	2%	2%	1181	< 0.5	< 0.5	6	6
2	14	494	516	501	508	504	3%	1%	7328	< 0.5	< 0.5	8	8
4	2	163	163	163	163	163	0%	0%	< 0.01	< 0.5	< 0.5	0	0
4	3	211	211	211	211	211	0%	0%	< 0.01	< 0.5	< 0.5	0	0
4	4	243	249	243	243	243	2%	0%	< 0.01	< 0.5	< 0.5	1	1
4	5	270	283	270	270	270	5%	0%	< 0.01	< 0.5	< 0.5	2	2
4	6	304	321	304	310	304	6%	2%	< 0.01	< 0.5	< 0.5	3	3
4	7	350	366	350	356	350	5%	2%	< 0.01	< 0.5	< 0.5	3	4
4	8	384	413	384	395	384	8%	3%	< 0.01	< 0.5	< 0.5	5	5
4	9	408	437	408	415	408	7%	2%	< 0.01	< 0.5	< 0.5	6	6
4	10	441	483	449	454	449	8%	1%	2	< 0.5	1	7	7
4	11	472	506	480	492	482	5%	2%	9	< 0.5	1	8	9
4	12	511	544	524	531	522	4%	2%	69	< 0.5	< 0.5	11	11
4	13	534	576	551	552	541	5%	2%	303	< 0.5	< 0.5	11	11
4	14	564	620	583	582	575	6%	1%	1105	< 0.5	< 0.5	14	14
6	2	211	211	211	211	211	0%	0%	< 0.01	< 0.5	< 0.5	< 0.5	< 0.5
6	3	257	257	257	257	257	0%	0%	< 0.01	< 0.5	< 0.5	< 0.5	< 0.5
6	4	302	311	302	302	302	3%	0%	< 0.01	< 0.5	< 0.5	1	1
6	5	331	347	331	332	331	5%	0%	< 0.01	< 0.5	< 0.5	3	3
6	6	372	389	372	385	372	5%	3%	< 0.01	< 0.5	< 0.5	4	4
6	7	417	446	417	429	417	7%	3%	< 0.01	< 0.5	< 0.5	5	5
6	8	448	473	448	463	448	6%	3%	< 0.01	< 0.5	< 0.5	6	7
6	9	479	509	479	493	479	6%	3%	< 0.01	< 0.5	< 0.5	8	8
6	10	516	566	529	531	522	7%	2%	2	< 0.5	1	10	10
6	11	549	593	559	569	558	6%	2%	10	< 0.5	1	12	12
6	12	583	627	601	608	597	4%	2%	57	< 0.5	1	14	15
6	13	626	669	642	653	640	4%	2%	553	< 0.5	1	17	18
6	14	660	724	679	680	674	7%	1%	4361	< 0.5	1	23	24

Table 2
Comparison of heuristic algorithms for $m = 2$.

n	$C_{\max,x}$				$\delta_{C,1}$	$\delta_{C,2}$	T_x			
	Alg1	AlgI1	Alg2	AlgI2			Alg1	AlgI1	Alg2	AlgI2
20	717	702	706	699	2%	1%	< 0.5	1	16	17
30	1053	1034	1057	1039	2%	2%	< 0.5	1	40	41
40	1383	1359	1400	1368	2%	2%	< 0.5	2	84	85
50	1740	1708	1769	1722	2%	3%	< 0.5	3	175	177
60	2073	2043	2135	2064	1%	3%	< 0.5	3	195	197
70	2398	2369	2479	2393	1%	4%	< 0.5	3	263	266
80	2754	2715	2859	2756	1%	4%	< 0.5	4	366	369
90	3056	3026	3207	3079	1%	4%	< 0.5	5	476	481
100	3385	3355	3542	3402	1%	4%	< 0.5	6	694	700

Table 3
Comparison of heuristic algorithms for $m = 4$.

n	$C_{\max,x}$				$\delta_{C,1}$	$\delta_{C,2}$	T_x			
	Alg1	AlgI1	Alg2	AlgI2			Alg1	AlgI1	Alg2	AlgI2
20	842	799	793	783	5%	1%	< 0.5	1	42	43
30	1194	1139	1154	1125	5%	3%	< 0.5	2	94	96
40	1555	1473	1497	1463	6%	2%	< 0.5	3	180	183
50	1890	1831	1875	1819	3%	3%	< 0.5	4	275	278
60	2244	2164	2250	2176	4%	3%	< 0.5	6	427	432
70	2601	2519	2628	2531	3%	4%	< 0.5	6	582	587
80	2952	2844	2994	2892	4%	4%	< 0.5	7	977	984
90	3265	3156	3338	3195	3%	4%	< 0.5	9	1167	1176
100	3576	3489	3684	3521	2%	5%	< 0.5	11	1808	1818

Table 4
Comparison of heuristic algorithms for $m = 6$.

n	$C_{\max,x}$				$\delta_{C,1}$	$\delta_{C,2}$	T_x			
	Alg1	AlgI1	Alg2	AlgI2			Alg1	AlgI1	Alg2	AlgI2
20	955	902	898	883	6%	2%	< 0.5	2	62	63
30	1366	1277	1287	1255	7%	3%	< 0.5	3	152	154
40	1738	1632	1657	1615	6%	3%	< 0.5	4	282	286
50	2074	2000	2047	1975	4%	4%	< 0.5	7	427	432
60	2398	2304	2404	2313	4%	4%	< 0.5	8	679	686
70	2813	2694	2819	2687	4%	5%	< 0.5	9	849	856
80	3139	3028	3206	3067	4%	5%	< 0.5	10	1430	1440
90	3463	3359	3573	3398	3%	5%	0	12	1637	1648
100	3803	3705	3927	3729	3%	5%	0	14	2532	2545

Some conclusions can be drawn from Table 1. Alg2 is better than Alg1 for the majority of tested instances. However, the computational time required by the former algorithm is slightly greater. The IP gives up to 8% enhancement in terms of the criterion $C_{\max}$ (Alg1 for $m = 4$, $n = 8$). The mean enhancement for Alg1 and Alg2 is equal to 3.9% and 1.5%, respectively. The profit of applying IP is gained at the cost of the very slight increase of the computation times. Alg1 and Alg2 are worse than BB up to 9.9% (Alg1 for $m = 4$, $n = 14$) as well as 4.7% and 2.3% worse on average, respectively. The computational times of all heuristic algorithms do not exceed 24 seconds.

The analysis of results from Tables 2–4, i.e. for greater instances, enables us to formulate the following conclusions. Alg1 and Alg2 return comparable results; the advantage of one of them is not noticed. The IP improves Alg1 and Alg2 up to 6%. The average improvement of Alg1 and Alg2 is equal to 4.2% and 4.4%, respectively. The computational times of Alg2 are considerably longer in comparison

with Alg1. The application of IP does not substantially extend them. All these times are still acceptable for real-world applications.

Case study

The proposed solution algorithms for the considered routing flow-shop can be used for the variety applications, especially in manufacturing, logistics and production systems. They can be also applied for complex systems when machines have the wider meaning going beyond technological facilities as well as the first parts of operations, i.e. driving-ups are not directly connected with the movement. The case study concerns such more general application. Let us consider a computer software company maintaining software products used by its clients. Such products need current repair, updating and can be developed by the company upon clients' requests or on its own initiative. Every such job requires a number of different specialists. A current problem in the company consists in the management of a given sets of jobs and specialists to minimize the total execution time, which is strictly connected with the maximization of the company's profit. It turned out that the described management problem can be modelled as the routing flow-shop to minimize the makespan. Each job J_j contains four following operations:

$O_{1,j}$ – detection of faults in a software product or receiving a corresponding maintenance request from clients,

$O_{2,j}$ – verification of the request by a manager and launching of successive operations,

$O_{3,j}$ – carrying out of the necessary changes and improvements in a code;

$O_{4,j}$ – testing of corrected software.

Four workers are involved in this maintenance process: auxiliary worker ($i = 1$), manager ($i = 2$), computer programmer ($i = 3$) and software tester ($i = 4$). Each job requires the preparation phase which can be only followed by the appropriate activity. This preparation phase corresponds to the driving-up times in the problem considered and can be caused e.g. by: preparation of necessary libraries, setting-up of the software environment, preparation of data bases. Moreover, while starting the maintenance process, some jobs can be not ready due to different technical reasons at the client's side that can be conveniently modelled via non-zero ready times.

Examples of the driving-up times and the execution times of activities for $n = 4$ are given in Tables 5–8. The following ready times were assumed: $r_1 = 65$, $r_2 = 110$, $r_3 = 63$, $r_4 = 113$. The management problem consists in the determination of permutation of jobs for which the total execution time of all given jobs is minimal. It is obvious that solutions returned by the algorithms with the improvement procedure were optimal which means that $\Pi^* = (J_0, J_1, J_4, J_2, J_3, J_5)$, and $C_{\max}(\Pi^*) = 2044$. The TS based heuristic algorithm Alg2 returned also the optimal solution, which was easy attainable for the problem consisting of four jobs, but the TSP based heuristic algorithm Alg1 returned the worse solution: $\Pi = (J_0, J_1, J_3, J_4, J_2, J_5)$, and $C_{\max}(\Pi) = 2083$.

Table 5
Driving-up times $\widehat{p}_{i,l,j}$ for $i = 1, 3$.

$l \setminus j$	1	2	3	4	5 (depot)
1	0	140	144	96	52
2	140	0	148	52	100
3	144	148	0	124	180
4	96	52	124	0	104
5 (depot)	52	100	180	104	0

Table 6
Driving-up times $\widehat{p}_{i,l,j}$ for $i = 2, 4$.

$l \setminus j$	1	2	3	4	5 (depot)
1	0	35	36	24	13
2	35	0	37	13	25
3	36	37	0	31	45
4	24	13	31	0	26
5 (depot)	13	25	45	26	0

Table 7
Execution times of activities $p_{i,j}$.

$i \setminus j$	1	2	3	4	5 (depot)
1	95	388	291	345	0
2	278	324	257	267	0
3	52	131	63	157	0
4	352	99	199	217	0

Conclusions

The selected routing flow-shop problem is considered in the paper. The permutation version with unlimited buffers as well as non-zero ready times and different speeds of machines is investigated. Due to the NP-hardness of the problem, the heuristic solution algorithms were the main subject of researches. Four particular algorithms have been proposed and evaluated via computer simulation. The algorithm referred to as AlgI2 is recommended for further applications. It is the hybrid algorithm composed of the TS metaheuristics and the Improvement Procedure which can advance TS by applying the branch

and bound procedure for selected small parts of the problem.

Further works will be focused on other versions of the problem, in particular on the case with the sum of completion times as the task scheduling criterion.

References

[1] Nagy G., Salhi S., *Location-routing: Issues, models and methods*, European Journal of Operational Research, 177, 649–672, 2007.

[2] Prodhon C., *A hybrid evolutionary algorithm for the periodic location-routing problem*, European Journal of Operational Research, 210, 204–212, 2011.

[3] Hamacher H.W., Hennes H., *Integrated scheduling and location models: single machine makespan problems*, Studies in Locational Analysis, 16, 77–90, 2007.

[4] Kalsch M.T., Drezner Z., *Solving scheduling and location problems in the plane simultaneously*, Computerts & Operations Research, 37, 256–264, 2010.

[5] Drezner Z., Scott C., Song, J.-S., *The central warehouse location problem revisited*, IMA Journal of Management Mathematics, 14, 321–336, 2003.

[6] Li J., Chen H., Chu F., *Performance evaluation of distribution strategies for the inventory routing problem*, European Journal of Operational Research, 202, 412–419, 2010.

[7] Moin N.H., Salhi S., *Inventory routing problems: A logistical overview*, Journal of the Operational Research Society, 58, 1185–1194, 2007.

[8] Józefczyk J., *Decision making problems in complex of operations systems* (in Polish), Wroclaw, Wroclaw University of Technology Press, 2001.

[9] Józefczyk J., *Scheduling tasks on moving executors to minimize the maximum lateness*, European Journal of Operational Research, 131, 171–187, 2001.

[10] Averbakh I., Berman O., *A simple heuristic for m-machine flow-shop and its applications in routing-scheduling problems*, Operations Research, 47, 165–170, 1999.

[11] Metters R.D., *Interdependent transportation and production activity at the United States postal service*, Journal of the Operational Research Society, 47, 27–37, 1996.

[12] Ben-Daya M., As'ad R., Seliaman M., *An integrated production inventory model with raw material replenishment considerations in a three layer supply chain*, International Journal of Production Economics, 143, 53–61, 2013.

[13] Filcek G., Józefczyk J., *Managing production and transportation in balanced supply networks*, Systems Science, 35, 49–59, 2009.

[14] Allahverdi A., Ng C.T., Cheng T.C.E., Kovalyov M.Y., *A survey of scheduling problems with setup times or costs*, European Journal of Operational Research, 187, 985–1032, 2008.

[15] Ruiz R., Allahverdi A., *No-wait flow shop with separate setup times to minimize maximum lateness*, International Journal of Advanced Manufacturing Technology, 35, 551–565, 2007.

[16] Ruiz R., Stutzle T., *An iterated greedy heuristic for the sequence dependent setup times flow shop problem with makespan and weighted tardiness objectives*, IRIDIA Technical Report, no. TR/IRIDIA/2006-02, 2006.

[17] Rajendran C.H., Ziegler H., *Scheduling to minimize the sum of weighted flow time and weighted tardiness of jobs in a flow shop with sequence – dependent setup times*, European Journal of Operational Research, 149, 513–522, 2003.

[18] Potts Ch.N., Kovalyov M.Y., *Scheduling with batching: a review*, European Journal of Operational Research, 120, 228–249, 2000.

[19] Thomas W., *Algorithms for task scheduling with moving executors to minimize the mean flow time* [in Polish], PhD Thesis, Technical Report of the Faculty of Computer Science and Management, Wroclaw University of Technology, PRE/2, 2006.

[20] Józefczyk J., Markowski M.: *Integrated optimization problems in operations research*, Analysis and decision making for complex and uncertain systems, Proceedings of 23^{rd} Int. Conf. on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, 1, 39–43, 2011.

[21] Yu W., Liu Z., Wang L., Fan T., *Routing open shop and flow shop scheduling problems*, European Journal of Operational Research, 213, 24–36, 2011.

[22] Averbakh I., Berman O., Chernykh I., *A 6/5 – approximation algorithm for the two-machine routing open-shop problem on a two-node network*, European Journal of Operational Research, 166, 3–24, 2005.

[23] Józefczyk J., Markowski M., *Heuristic algorithms for solving uncertain routing-scheduling problem*, Lecture Notes in Computer Science, 5097, 1052–1063, 2008.

[24] Józefczyk J., Markowski M., *Simulated annealing based robust algorithm for routing-scheduling problem with uncertain execution times*, Proceedings of 13th IFAC Symposium on Information Control Problems in Manufacturing, INCOM '09, Moscow, Russia, pp. 1985–1990, 2009.

[25] Cheng J., Steiner G. Stephenson P., *A computational study with a new algorithm for the three-machine permutation flow-shop problem with release times*, European Journal of Operational Research, 130, 559–575, 2001.

[26] Framinan J.M., Gupta J.N.D., Leisten R., *A review and classification of heuristics for permutation flow-shop scheduling with makespan objective*, Journal of the Operational Research Society, 55, 1243–1255, 2004.

[27] Rebaine D., *Flow shop vs. permutation shop with time delays*, Computers & Industrial Engineering, 48, 357–362, 2005.

[28] Pinedo M.L., *Scheduling theory, algorithms and systems*, New York, Springer, 2008.

[29] Józefczyk J., Markowski M., *Heuristic solution algorithm for routing flow shop with buffers and ready times*, Analysis and decision making for complex and uncertain systems, Proceedings of 24^{th} Int. Conf. on Systems Research, Informatics and Cybernetics, Baden-Baden, Germany, 2, 37–41, 2012.

[30] Golden B., Bodin L., Doyle T., Stewart W. Jr., *Approximate traveling salesman algorithms*, Operational Research, 28, 3, Part II, 694–711, 1980.