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

Jerzy Józefczyk¹, Michał Markowski¹, Ljazat Balgabaeva²

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

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, queuing, 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-
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 $M = \{M_1, M_2, \ldots, M_m\}$ and $J = \{J_1, J_2, \ldots, J_n\}$ 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 $\mathbf{J} = \{J_1, \ldots, J_n, J_{n+1}\}$. 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 $p_{i,t,j}$ and $p_{i,j}$ the driving-up time of machine $M_i$ from workstation $J_t$ to workstation $J_j$, and the execution time of activity $O_{i,j}$, respectively. In a consequence, $\tilde{p}_{i,t,j} = p_{i,j} + p_{i,t,j}$ 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 $\Pi = (J_{\pi_0}, J_{\pi_1}, \ldots, J_{\pi_{n+1}}) \in \Pi$ where $\Pi$ 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 $\Pi = (J_{\pi_1}, J_{\pi_2}, J_{\pi_3}, J_{\pi_4})$, $(\pi_0, \pi_1, \pi_2, \pi_3, \pi_4, \pi_5) = (5, 3, 4, 2, 1, 5)$ is presented in Fig. 1.

![Fig. 1. Example of a layout of workstations and machines’ routes.](image)

Problems formulation

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

$$C(\Pi, i, k) = \max\{C(\Pi, i, k-1) + \tilde{p}_{i,\pi_{k-2},\pi_{k-1}}, C(\Pi, i-1, k) + \tilde{p}_{i-1,\pi_{k-1},\pi_k} - \tilde{p}_{i,\pi_{k-1},\pi_k}\}$$

(1)

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

$$C(\Pi, i, 1) = \max\{C(\Pi, i-1, 1) + \tilde{p}_{i-1,\pi_0,\pi_1} - \tilde{p}_{i,\pi_0,\pi_1}, 0\}$$

(2)

for $k = 1$.

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

$$C(\Pi, 1, k) = \max\{C(\Pi, 1, k-1) + \tilde{p}_{1,\pi_{k-2},\pi_{k-1}}; r_{\pi_k} - \tilde{p}_{1,\pi_{k-1},\pi_k}\}$$

(3)

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

$$C(\Pi, 1, 1) = \max\{0; r_{\pi_1} - \tilde{p}_{1,\pi_0,\pi_1}\}.$$  

(4)

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

$$C_{\text{max}}(\Pi) = \max_{i=1,2,\ldots,m} \{C(\Pi, i, n) + \tilde{p}_{i,\pi_{n-1},\pi_n} + \tilde{p}_{i,\pi_n,\pi_{n+1}}\}.$$  

(5)

So, the considered routing flow-shop problem consists in the determination of such sequence $\Pi$ to
minimize $C_{\text{max}}(\Pi)$ for given: $M, J, \pi_i,j, \tilde{p}_{i,j}, r_j$, $i = 1, 2, \ldots, m$, $k = 1, 2, \ldots, n$.

As the result, $\Pi^*$ and $C_{\text{max}}(\Pi^*)$ are obtained. This optimization problem is at least NP-hard due to the NP-hardness of its 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 Alg1I and Alg1II, respectively.

Branch and bound exact algorithm

The branching procedure enables us checking all feasible permutations $\Pi$, 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 $C_{\text{max}}$ referred to as $C_{\text{LB}}$ are calculated at every vertex, and they make possible to limit the searching procedure along the tree. To perform the $\Pi$ partial solution (permutation) ready at vertex $v$, which contains jobs belonging to the set $J(v) \subset J$. The jobs not scheduled yet form the set $\mathcal{J}(v) = J \setminus J(v)$. Then, jobs from sets $J(v)$ and $\mathcal{J}(v)$ belong to sub-solution $\Pi(v) = (J_{n+1} \cup J_{\pi_1} \cup \ldots \cup J_{\pi_l} \cup \ldots \cup J_{\pi_{|J(v)|}})$, $J_{\pi_l} \in \mathcal{J}(v)$ and $\Pi'(v) = \Pi \setminus \Pi(v)$, respectively.

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

$$
C_{\text{LB}}(\Pi, k) = \max_{i = 1, 2, \ldots, m} [C(\Pi, i, k + 1)] + \sum_{J_{\pi_l} \notin \mathcal{J}(v)} \tilde{p}_{m, \pi_l, \pi_l + 1} + \sum_{J_{\pi_l} \notin \mathcal{J}(v)} \tilde{p}_{m, \pi_l, \pi_l + 1} + \min_{J_{\pi_l} \in \mathcal{J}(v)} \tilde{p}_{m, \pi_l, \pi_l + 1} + \max[0];
$$

$$
\min_{J_{\pi_l} \in \mathcal{J}(v)} r_{\pi_l} - \max_{i = 1, 2, \ldots, m} (C(\Pi, i, k + 1)) - \max_{J_{\pi_l} \in \mathcal{J}(v)} \tilde{p}_{m, \pi_l, \pi_l + 1},
$$

and

$$
p_{\text{min}}^{\text{\Pi}} = \min_{J_{\pi_l} \notin \mathcal{J}(v)} \tilde{p}_{m, \pi_l, \pi_l}, \quad p_{\text{max}} = \max_{J_{\pi_l} \in \mathcal{J}(v)} p_{\pi_l}^k.
$$

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 $J'(v)$, the sum of minimum driving-up times to every remaining job from $J'(v)$ adjusted by $p_{\text{max}}$, the minimum of driving-up times from locations of jobs belonging to $J'(v)$ 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_1}, J_{\pi_2}, \ldots, J_{\pi_n}) \varepsilon - \text{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 $\Pi_\varepsilon$), $\Pi_\varepsilon^T = (J_{\pi_{n+1}}, J_{\pi_n}, \ldots, J_{\pi_1})$ - the sequence of workstations reverse to $\pi_\varepsilon$, $\mathcal{T}$ - result of Alg1. Then, Alg1 is composed of two steps.

Algorithm Alg1

Input: algorithm for TSP returning the solution $\Pi_\varepsilon$.

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

1. Determine $\Pi_\varepsilon$ and $\Pi_\varepsilon^T$, as well as calculate $C_{\text{max}}(\Pi_\varepsilon)$, and $C_{\text{max}}(\Pi_\varepsilon^T)$.

2. If $C_{\text{max}}(\Pi_\varepsilon) \leq C_{\text{max}}(\Pi_\varepsilon^T)$ set $\Pi = \Pi_\varepsilon$, otherwise set $\Pi = \Pi_\varepsilon^T$, and calculate $C_{\text{max}}(\Pi)$.

Any known $\varepsilon$ – approximation algorithm solving TSP can be used to obtain $\Pi_\varepsilon$. The computational complexity of Alg1 is determined by the used $\varepsilon$ – 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 $v_{\pi_s, \pi_k}(\Pi) \in \Pi$ denotes the replacement of positions between jobs $\pi_s$ and $\pi_k$ in solution $\Pi$. The new solution $v_{\pi_s, \pi_k}(\Pi)$ is obtained as the result, which can denote also another solution generated after performing the move. The moves on $\Pi$ constitute the $\mathcal{N}$ – element set $\mathcal{V}(\Pi)$ called also the neighborhood of $\Pi$.

The tabu list $\mathcal{T}(T_1, T_2, \ldots, T_L)$ of $L$ elements $T_i$ contains attributes of solutions and moves referred to as $a_{\mathcal{V}_s}(\Pi, v) \in \mathcal{A}_s(\Pi, v)$ where $\mathcal{A}_s(\Pi, v)$ is the set of attributes defined in the paper as $A(\Pi, v) = \{\pi_{i-1, i}, (\pi_{i, i+1}), i = 1, 2, \ldots, n\}$. All moves for $\Pi$ are divided into forbidden $\mathcal{V}_s(\Pi) \subseteq \mathcal{V}(\Pi)$ and

Volume 5 - Number 4 - December 2014
free \( V_1(II) \subseteq V(II) \) ones, i.e. \( V_1(II) \cup V_2(II) = V(II) \). The move for which \( C_{\text{max}}(v_{\pi_i, \pi_k}(II)) \leq C_{\text{max}}(\tilde{\Pi}) \) holds, where \( C_{\text{max}}(\tilde{\Pi}) \) is the makespan for the current best solution \( \tilde{\Pi} \) generated by Alg2, is called the prospective forbidden move and is an element of the subset \( V_2(II) \subseteq V(II) \) of prospective forbidden moves. Other elements of \( V_2(II) \) form a subset \( V_2(II) = V_2(II) \setminus V_2(II) \) which comprises non-prospective forbidden moves. Alg2 starts with initial population \( \Pi_0 \) generated randomly or being the result of another heuristic algorithm and ends after \( N \) iterations.

**Algorithm Alg2**

**Input:** \( TL = \emptyset, \kappa = 1, \tilde{\Pi} = \Pi = \Pi_0, \pi, N \)

**Output:** \( \tilde{\Pi}, C_{\text{max}}(\tilde{\Pi}) \)

1. Determine \( V(II) \) performing \( \pi \) times the moves \( v_{\pi, \pi_k}(II) \), where \( \pi, \pi_k \) are randomly generated indices of jobs.
2. Determine sets \( V_1(II), V_2(II), V_2(II) \), and go to Step 4 if \( V_1(II) \cup V_2(II) = \emptyset \).
3. Find the move \( v_{\pi, \pi_k}(II) \) minimizing \( C_{\text{max}}(\Pi) \) i.e. \( C_{\text{max}}(v_{\pi, \pi_k}(II)) = \min C_{\text{max}}(\Pi) \) and go to Step 5.
4. If \( |V_2(II)| = 1 \) denote element of \( V_2(II) \) as \( v_{\pi, \pi_k}(II) \) and go to Step 6 repeat else repeat Steps 4a and 4b until \( V_1(II) \cup V_2(II) \neq \emptyset \) or \( T_L = T_L, l = 1, 2, ..., L \):
   a. Set \( TL = TL \oplus T_L \), i.e. join \( TL \) to the end of \( TL \).
   b. Determine new sets \( V_1(II), V_2(II), V_2(II) \) and go to Step 3.
5. Set \( TL = TL \oplus a_\pi(II, v_{\pi, \pi_k}(III)) \), \( \Pi = v_{\pi, \pi_k}(II) \).
6. Calculate \( C_{\text{max}}(\Pi) \).
7. If \( C_{\text{max}}(\Pi) < C_{\text{max}}(\tilde{\Pi}) \) set: \( \tilde{\Pi} = \Pi, C_{\text{max}}(\tilde{\Pi}) = C_{\text{max}}(\Pi), \kappa = 1 \).
8. 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 \( \tilde{\Pi} \) obtained by Alg2 (the case of \( \Pi \) produced by Alg1 follows accordingly). More precisely, B&B is simultaneously used for sub-permutations \( \tilde{\Pi}_\lambda \) of \( \tilde{\Pi} \) where \( \tilde{\Pi} = (\tilde{\Pi}_1, ..., \tilde{\Pi}_\lambda, ..., \tilde{\Pi}_\Lambda) \), and \( \Lambda = \lfloor n/\eta \rfloor \) is the number of sub-permutations of the length \( n \). Every sub-permutation consists of a part of \( \tilde{\Pi} \), i.e.

\[
\tilde{\Pi}_\lambda = (J_0, J_{\pi(\lambda-1)+1}, J_{\pi(\lambda-1)+2}, ..., J_{\lambda n}, J_{n+1}), \lambda = 1, 2, ..., \Lambda - 1,
\]

\[
\tilde{\Pi}_\Lambda = (J_0, J_{\pi(\Lambda-1)+1}, J_{\pi(\Lambda-1)+2}, ..., J_{\Lambda n}, J_{n+1}).
\]

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:** \( \tilde{\Pi}, \eta \)

**Output:** \( \tilde{\Pi}_{BB}, C_{\text{max}}(\tilde{\Pi}_{BB}) \)

1. Divide permutation \( \tilde{\Pi} \) into \( \Lambda \) sub-permutations \( \tilde{\Pi}_1, \tilde{\Pi}_2, ..., \tilde{\Pi}_\lambda, ..., \tilde{\Pi}_\Lambda \).
2. Repeat for \( \lambda = 1, 2, ..., \Lambda \):
   a. Determine new sets \( \Pi_{\lambda}, \Pi_{\lambda+1} \) until \( \Pi = \Pi_{\lambda+1} \).
   b. Determine new sets \( \Pi_{\lambda}, \Pi_{\lambda+1} \) until \( \Pi = \Pi_{\lambda+1} \).
   c. Compose the final solution (after excluding excessive depots necessary for partial solutions) as \( \tilde{\Pi}_{BB} = (J_0, \tilde{\Pi}_{BB}, \tilde{\Pi}_{BB}, \tilde{\Pi}_{BB}, \tilde{\Pi}_{BB}, \tilde{\Pi}_{BB}), \lambda = 1, 2, ..., \Lambda - 1, \)
   d. Calculate \( C_{\text{max}}(\tilde{\Pi}_{BB}) \).

**Hybrid algorithms**

When launching the improvement procedure with \( \tilde{\Pi} \), being the result of Alg1, as the initial solution, we have hybrid algorithm Alg11. Analogously, merging of Alg2 with the improvement procedure gives Alg12 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: \( p_{ij}, p_{i,j} \in [1, 2, ..., 50], r_j \in [0, 1, 20] \). The algorithms were launched for the following parameters: Alg1 – the cheapest insertion method as the algorithm solving TSP; Alg2 – \( \pi = 6 \), \( N = 6000 \) and \( \Pi_0 \) 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 $C_{\text{max}}$ and the execution times $T_x$ are the basis for evaluation. The notation is proposed to distinguish the algorithms: $C_{\text{max},x}$, $T_x$, $x \in \{\text{BB, Alg1, Alg11, Alg2, AlgI2}\}$. 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 $C_{\text{max}}$ caused by the application of IP in Alg1 and Alg2 are also inserted, where $\delta_{C,1} = \frac{C_{\text{max},\text{Alg1}} - C_{\text{max},\text{Alg11}}}{C_{\text{max},\text{Alg11}}} \cdot 100\%$ and $\delta_{C,2} = \frac{C_{\text{max},\text{Alg2}} - C_{\text{max},\text{AlgI2}}}{C_{\text{max},\text{AlgI2}}} \cdot 100\%$.

Table 1
Comparison of heuristic and exact algorithms.

<table>
<thead>
<tr>
<th>$m$</th>
<th>$n$</th>
<th>$C_{\text{max},x}$</th>
<th>$\delta_{C,1}$</th>
<th>$\delta_{C,2}$</th>
<th>$T_x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>BB 105</td>
<td>0%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>Alg1 150</td>
<td>0%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>Alg2 178</td>
<td>4%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>AlgI2 206</td>
<td>1%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>BB 239</td>
<td>3%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>Alg1 274</td>
<td>3%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>Alg2 306</td>
<td>4%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>AlgI2 336</td>
<td>4%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>BB 369</td>
<td>2%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>Alg1 400</td>
<td>2%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>Alg2 421</td>
<td>2%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>2</td>
<td>13</td>
<td>AlgI2 461</td>
<td>2%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>BB 494</td>
<td>3%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>Alg1 163</td>
<td>0%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>Alg2 211</td>
<td>0%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>AlgI2 243</td>
<td>2%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>BB 270</td>
<td>5%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>Alg1 304</td>
<td>6%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>Alg2 350</td>
<td>5%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>AlgI2 384</td>
<td>8%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>BB 408</td>
<td>7%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>Alg1 441</td>
<td>8%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>4</td>
<td>11</td>
<td>Alg2 472</td>
<td>5%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>AlgI2 511</td>
<td>4%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>4</td>
<td>13</td>
<td>BB 534</td>
<td>5%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>Alg1 564</td>
<td>6%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>Alg2 211</td>
<td>0%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>AlgI2 257</td>
<td>0%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>BB 302</td>
<td>3%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>Alg1 332</td>
<td>5%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>Alg2 372</td>
<td>5%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>AlgI2 417</td>
<td>7%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>BB 448</td>
<td>6%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>6</td>
<td>9</td>
<td>Alg1 479</td>
<td>6%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>6</td>
<td>10</td>
<td>Alg2 516</td>
<td>7%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>6</td>
<td>11</td>
<td>AlgI2 549</td>
<td>6%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>BB 583</td>
<td>4%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>6</td>
<td>13</td>
<td>Alg1 626</td>
<td>4%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>6</td>
<td>14</td>
<td>Alg2 660</td>
<td>4%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
<tr>
<td>6</td>
<td>15</td>
<td>AlgI2 692</td>
<td>7%</td>
<td>&lt; 0.01</td>
<td>&lt; 0.5</td>
</tr>
</tbody>
</table>

Volume 5 • Number 4 • December 2014
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 \( C_{\text{max}} \) (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 computational 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 \( H^* = (J_0, J_1, J_4, J_2, J_3, J_5) \), and \( C_{\text{max}}(H^*) = 2044 \). 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: \( H = (J_0, J_1, J_3, J_4, J_2, J_5) \), and \( C_{\text{max}}(H) = 2083 \).

### Table 5
Driving-up times \( \hat{p}_{i,l,j} \) for \( i = 1, 3 \).

<table>
<thead>
<tr>
<th>( i ) ( j )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5 (depot)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>140</td>
<td>141</td>
<td>96</td>
<td>52</td>
</tr>
<tr>
<td>2</td>
<td>140</td>
<td>0</td>
<td>148</td>
<td>52</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>144</td>
<td>148</td>
<td>0</td>
<td>124</td>
<td>180</td>
</tr>
<tr>
<td>4</td>
<td>96</td>
<td>52</td>
<td>124</td>
<td>0</td>
<td>104</td>
</tr>
<tr>
<td>5 (depot)</td>
<td>52</td>
<td>100</td>
<td>180</td>
<td>104</td>
<td>0</td>
</tr>
</tbody>
</table>

### Table 6
Driving-up times \( \hat{p}_{i,l,j} \) for \( i = 2, 4 \).

<table>
<thead>
<tr>
<th>( i ) ( j )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5 (depot)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>35</td>
<td>36</td>
<td>24</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>35</td>
<td>0</td>
<td>37</td>
<td>13</td>
<td>25</td>
</tr>
<tr>
<td>3</td>
<td>36</td>
<td>37</td>
<td>0</td>
<td>31</td>
<td>45</td>
</tr>
<tr>
<td>4</td>
<td>24</td>
<td>13</td>
<td>31</td>
<td>0</td>
<td>26</td>
</tr>
<tr>
<td>5 (depot)</td>
<td>13</td>
<td>25</td>
<td>45</td>
<td>26</td>
<td>0</td>
</tr>
</tbody>
</table>

### Table 7
Execution times of activities \( p_{i,j} \).

<table>
<thead>
<tr>
<th>( i ) ( j )</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5 (depot)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>95</td>
<td>388</td>
<td>291</td>
<td>345</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>278</td>
<td>324</td>
<td>257</td>
<td>267</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>52</td>
<td>131</td>
<td>63</td>
<td>157</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>352</td>
<td>99</td>
<td>199</td>
<td>217</td>
<td>0</td>
</tr>
</tbody>
</table>

### 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


