JOURNAL OF CIVIL ENGINEERING AND MANAGEMENT
ISSN 1392-3730 / eISSN 1822-3605
2014 Volume 20(5): 649–659
doi:10.3846/13923730.2014.906496

SOLVING RESOURCE-CONSTRAINED CONSTRUCTION SCHEDULING PROBLEMS WITH OVERLAPS BY METAHEURISTIC

Wojciech BOŻEJKO[a], Zdzisław HEJDUCKI[b], Mariusz UCHROŃSKI[a], Mieczysław WODECKI[c]
[a]*Institute of Computer Engineering, Control and Robotics, Wrocław University of Technology, Janiszewskiego 11-17, 50-372 Wrocław, Poland*
[b]*Institute of Construction, Wrocław University of Technology, Wyb. Wyspiańskiego 27, 50-370 Wrocław, Poland*
[c]*Institute of Computer Science, University of Wrocław, Joliot-Curie 15, 50-383 Wrocław, Poland*

Received 25 Apr 2013; accepted 23 Sep 2013

Abstract. The paper concerns the problem of roadworks scheduling executed in the flow-shop system. Works may be performed parallelly with the acceleration (overlaps) of construction project, i.e. the following work on the assembly line can begin before the completion of the predecessor work. Taking into account the acceleration enables accurate modeling of complex real construction processes. The above fact can greatly shorten the time of realization of construction process which has a direct impact on reducing costs. The considered issue belongs to the class of NP-hard problems. We introduce the new: mathematical model, specific properties as an acceleration tools, as well as two new optimization algorithms for the problem considered: construction and tabu search. The execution of algorithms was illustrated on the example of a case study concerning the construction of roads. They were also verified on the examples taken from the literature and on already completed construction processes. The obtained results are fully satisfactory. The assigned execution times are close to optimal. The presented methods allow its practitioners (both the planners and the managers) to include in the model the acceleration of the works and the design of a much more efficient construction scheduling. The presented new scheduling method leads to a more competitive environment for contraction companies.

Keywords: scheduling, road building, flow shop, overlaps, tabu search.

Introduction

Construction project planning processes are usually carried out using techniques based on the critical path method (CPM) (Mattila, Park 2003). CPMs have evolved (from simple diagrams to sophisticated commercial computer programmes used today) for over 40 years. The network scheduling of continuous linear processes has been presented in many papers dealing with graphical network planning techniques (e.g. Lucko, PeñaOrozco 2009; Arditi *et al.* 2001; Hegazy 1999; Johnston 1984). Construction process synchronization problems have been solved for linear engineering structures (i.e. railways, motorways, etc.). Novel network planning methods have been proposed by Hamerlink and Rowings (1998) and Harris and Ioannou (1998). Solutions to many specific problems can be found in works by Chrzanowski and Johnston (1996), Harris and Ioannou (1998) and Zavadskas *et al.* (2010). Solutions of production scheduling are provided by Karwat (2012) in industrial application.

For linear scheduling methods applied to construction projects scheduling – deterministic (e.g. O'Brien 1969; Hamerlink, Rowings 1998; Mattila, Abraham 1998; Harris, Ionnou 1998; El-Rayes, Moselhi 2001) as well as fuzzy and probabilistic (e.g. Moselhi, Lorterapong 1995) – an optimization process is executed for a fixed order of tasks. In this paper we allow to change the order of the executed tasks, which situation takes place when a company performs construction projects (e.g. roadworks) on several working fronts parallelly. That is why we propose robust metaheuristics approximate methods for minimization of such construction projects make spans.

There is a number of applications designed to create construction projects schedules in literature. However, if the model is equal to *NP-hard* optimization problem, then existing algorithms for real-life data (hundreds of jobs) are not able to find an optimal solution in a reasonable time (computational complexity is exponential). Therefore, approximate algorithms are usually applied (Bożejko *et al.* 2012).

The development of optimization methods, particularly applied to resource-constrained scheduling, has proceeded towards modern and more effective sequence approaches since the beginning of this field. At the end of the 1970s, the turning point in the combinatorial optimization methods was the branch and bound (B&B) method regarded those days as a remedy for nearly all problems of great size which could not be solved by means of methods applied at that time. However, it soon occurred that

Corresponding author: Wojciech Bożejko
E-mail: wojciech.bozejko@pwr.wroc.pl

the B&B method only slightly extended the scope of solvable problems (e.g. for a sum-cost, single-machine scheduling problem this size extended from 20 to 40–50 tasks). What is more, the cost necessary to obtain an optimal solution is much too high compared to economic benefits and its use in practice. The conclusion of these investigations was the definition of a bounded area of the B&B scheme application.

The next breakthrough concerned the occurrence of advanced approximate methods: dedicated for a particular problem as well as metaheuristic (general schemas) – first the simulated annealing method and next the method of genetic algorithms and the tabu search method. Until around 2005 several dozen of different metaheuristics had been proposed (i.e. scatter search, ant colony optimization, particle swarm optimization, genetic programming, etc.) though again those methods reached the limit of their abilities to the moment where the size of effectively solvable problems (i.e. these for which an average deviation from the optimal solutions was smaller than 1%) might be shifted to a number reaching thousands, but not millions or hundred millions. Eventually the concept of "no-free-lunch" by Wolpert and Macready (1997) finished the discussion. With reference to rough methods this concept may be paraphrased in the following way: without using special attributes of examined problems considerable advantage of one metaheuristic over the other cannot be obtained. Nowadays, the classification of scheduling algorithms are: (1) analytical (exact) algorithms; (2) heuristic (dedicated) methods; and (3) metaheuristics.

In this paper we are using the newest achievements in this field applying algorithms which have proved to be very effective in solving classic optimization problems such as traveling salesman problem (TSP) or flow shop problem (Rogalska *et al.* 2008). We are proposing not only a description of the problem, but also a mathematical model and its solving algorithms. What is more, we prove specific properties (according to "no-free-lunch" theorem), significantly accelerating the calculations performed by algorithm based on tabu method. The whole procedure is illustrated on the example of a certain part of a road divided into several segments, on which the same works are performed in the same sequence (some of them start in advance).

1. Construction project problem with overlaps

We are considering the construction work (*Construction Project with Overlaps*, in short denoted by **CPwO**) involving execution of many construction projects. Each project consists of the same number of works to be done in a fixed sequence (the same for each object) set by the technological order. The works are performed by different brigades. There are data of work execution times and overlapping times given. The problem consists in determining the starting and ending dates of individual work performance on the projects, which minimize the completion date of all the projects that meet the following restrictions:

(i) each work can be performed only by a single, defined by a technological order, brigade;
(ii) neither brigade can perform at the same time more than one work;
(iii) for each project there must be a technological order maintained, i.e. the same order of work execution;
(iv) execution of any work cannot be interrupted before it is completed;
(v) any work can be started in advance (with overlapping), i.e. before the end of the one preceding it in the technological order;
(vi) any work cannot be completed before the end of the one preceding it in the technological order.

The optimization issues related to the scheduling of construction works are usually formulated as discrete optimization problems, namely, works scheduling on machines, where projects correspond to tasks, works correspond to operations and brigades to machines. In the further part of the paper we will use definitions and designations used in tasks scheduling theory.

The above considered construction project with overlaps (**CPwO** problem in short) can be defined as follows. There is a set of jobs:

$$J = \{J_1, J_2, ..., J_n\},$$

to be carried out using a set of machines:

$$M = \{M_1, M_2, ..., M_m\}.$$

Each job $J_i \in J$ consists of m indivisible operations $J_i = (O_{1,i}, O_{2,i}, ..., O_{m,i})$. The set of all operations:

$$O=\{O_{1,1}, O_{2,1},..,O_{m,1}, O_{1,2}, O_{2,2},..,O_{m,2}, \dots, O_{1,n}, O_{2,n},..,O_{m,n}\}.$$

Operation $O_{k,i}$ is to be performed on the machine M_k in the time $p_{k,i}$. For each $O_{k,i}$ operation there is defined the overlapping $l_{k,i}$ which means that $O_{k,i}$ overlaps operation $O_{k-1,i}$ ($l_{0,i} = 0$). A case $l_{k,i} > 0$ means the permission for 'overlapping' of subsequent job's operations, or the start of the next job's operation with some time delay compared to the start of the current operation and before its completion. We assume that the "overlapping" of the first operation of each task is equal to 0. We introduce the following variables:

$S_{i,j}, C_{i,j}$	–	searched due date of starting and ending operation $O_{i,j}$;
S	–	due date of all the operations start (without loss of generality we assume $S = 0$);
$C_{\max}$	–	searched date for ending of all operations.

Using the above designations, the considered building project can be described as the following optimization problem:

Minimize $$C_{\max}, \qquad (1)$$

subject to $S_{i,j} \geq 0,\ i=1,\ldots,m,\ j=1,\ldots,n;$ (2)

$$S_{i,j} \geq C_{i,j-1},\ i=2,\ldots,m,\ j=1,\ldots,n; \quad (3)$$

$$S_{i,j} \geq S_{i,j-1} + p_{i,j-1},\ i=1,\ldots,m,\ j=2,\ldots,n; \quad (4)$$

$$C_{i,j} = S_{i,j} + p_{i,j},\ i=1,\ldots,m,\ j=1,\ldots,n; \quad (5)$$

$$S_{i,j} \geq C_{i-1,j} - l_{i,j},\ i=2,\ldots,m,\ j=1,\ldots,n; \quad (6)$$

$$C_{i,j} \geq C_{i-1,j},\ i=2,\ldots,m,\ j=1,\ldots,n; \quad (7)$$

$$C_{i,j} \leq C_{\max},\ i=1,\ldots,m,\ j=1,\ldots,n. \quad (8)$$

Constraints (3) are a mathematical record (*i*), constraints (4) correspond to (*ii*), constraints (5) refer to (*iv*), while (6) corresponds to (*iii*) and (*v*) and the constraints (7) correspond to (*vi*). All other constraints are obvious. Any solution (acceptable) of **CPwO** problem is a sequence of *nm* numbers (the start dates of operations):

$$(S_{1,1}, S_{1,2}, \ldots, S_{1,n}, S_{2,1}, S_{2,2}, \ldots, S_{2,n}, \ldots, S_{i,1}, S_{i,2}, \ldots,$$
$$S_{i,n}, \ldots, S_{m,1}, S_{m,2}, \ldots, S_{m,n}),$$

satisfying the constraints (2)–(8). Such a solution can be represented by permutation $\pi = (\pi(1), \ldots, \pi(n))$ tasks from set *J*. It is easy to note that a non-decreasingly ordered set $\{S_{1,1}, S_{1,2}, \ldots, S_{1,n}\}$ of non-decreasing start times of tasks (i.e. the first operation of each task) on the first machine determines permutations of tasks (the same for each machine). On the other hand, for any permutation of the tasks π we can determine the starting times of operations using the following recursive dependencies:

$$S_{1,\pi(1)} = 0,\quad S_{1,\pi(i)} = S_{1,\pi(i-1)} + p_{1,\pi(i-1)},\ i=2,3,\ldots,n,$$
$$S_{j,\pi(1)} = S_{j-1,\pi(1)} + p_{j-1,\pi(1)} - l_{j,\pi(1)},\ j=2,3,\ldots,m,$$
$$S_{j,\pi(i)} = \max\{S_{j-1,\pi(i)} + p_{j-1,\pi(i)} - l_{j,\pi(i)}, S_{j,\pi(i-1)} + p_{j,\pi(i-1)}\},$$

for $i=2,3,\ldots,n,\ j=2,3,\ldots,m.$

Computational complexity of starting times calculated in the above manner equals $O(nm)$.

Remark 1. Date of completion of the tasks performed in the order π:

$$C_{\max}(\pi) = C_{m,\pi(n)}.$$

By Φ we denote the set of all permutations of the solution of tasks from J. The solution to the herein considered problem of construction project with overlaps **CPwO** (i.e. equivalent to problems of tasks scheduling defined by the constraints (1)–(8)) is to determine the optimal permutation $\pi^* \in \Phi$ such that:

$$C_{\max}(\pi^*) = \min\{C_{\max}(\pi) : \pi \in \Phi\}.$$

The problem which is presented in this section concerns construction projects scheduling in which – using the language of automation – jobs should start on the next machine before being finished on the previous one. In a classical permutation flow shop problem each of jobs should be carried out one after the other; moreover, the sequence of carrying out jobs on each machine has to be the same; start of carrying out a job on a next machine has not to follow until it is finished on the previous machine. Optimization consists in determining such a sequence of carrying out jobs that it will minimize the total times of carrying them out.

In the literature there were different generalizations and special cases of the problem **CPwO** considered. They cover both the objective function and the constraints. If we assume that the acceleration times (overlaps) $l_{i,j} = 0$ ($O_{i,j} \in O$) we get a classical permutation flow shop problem with the C_{max} criterion. In literature this problem is denoted by $F \mid perm \mid C_{\max}$ and is strongly *NP-hard* (from this fact it follows that **CPwO**problem is also *N*P-*hard*). Although for over 30 years intensively examined permutational flow problem is already classical optimization problem nowadays, however there are still no satisfactory algorithms of its solving, in particular for practical instances of big sizes. Instances which are not bigger than 20 jobs and 5 machines can be solved, in a reasonable time, by exact algorithms based on the branch and bound method. In last years many algorithms of the $F \mid perm \mid C_{\max}$ problem based on tabu search method (Grabowski, Wodecki 2004), genetic algorithm (Reeves, Yamada 1998; Bożejko, Wodecki 2004), parallel algorithm, scatter search (Bożejko, Wodecki 2008) and simulated annealing (Ishubuchi *et al.* 1995), parallel algorithm (Wodecki, Bożejko 2002) have been published. Generally, the quality of defined by these algorithms solutions is dependent, to a large extent, on their operation time. The higher the time, the bigger subset of the set of acceptable solutions is searched (directly), which increases the opportunity to find a good solution.

Many works are also devoted to the flow-shop problem with transportation time (i.e. transport times of jobs between machines). Work of Gupta and Stafford (2006) is a classical study of the problem. The latest results and literature review are presented in the works of Gupta (2012) and Lamoudan *et al.* (2011).

2. Graph model

For any feasible solution, i.e. the order of execution of tasks on machines (permutations) $\pi = (\pi(1), \ldots, \pi(n))$, we construct a directed graph with burdened vertices and arcs $G(\pi) = (V, A(\pi))$, where a set of vertices $V = O$ (vertices correspond to operations), and $A(\pi) = R \cup E(\pi)$ and a set of arcs, where:

1. $$R = \bigcup_{j=1}^{n} \left[\bigcup_{i=1}^{m-1} \left(O_{i,\pi(j)}, O_{i+1,\pi(j)} \right) \right].$$

The set *R* contains arcs connecting consecutive operations of the same task (they represent the technological order). These are called *vertical arcs*;

2. $$E(\pi)=\bigcup_{k=1}^{m}\bigcup_{i=1}^{n-1}\left\{\left(O_{k,\pi(i)},O_{k,\pi(i+1)}\right)\right\}.$$

Arcs from the set $E(\pi)$ combine the operations performed by the machine, setting the sequence of tasks (the same on each machine). These are called *horizontal arcs*.

Vertices and arcs of the graph are loaded with weights. Weight of vertex $O_{i,j}$ is equal to the time of operation execution $p_{i,j}$. The weight of vertical arc $\left(O_{i-1,j},O_{i,j}\right)\in R$ is $-l_{i,j}$ (i.e. minus overlap). The weights of all other arc (horizontal) equal 0. Figure 1 shows part of the graph $G(\pi)$, $\pi\in\Phi$ (without marked burdens of vertices and arcs).

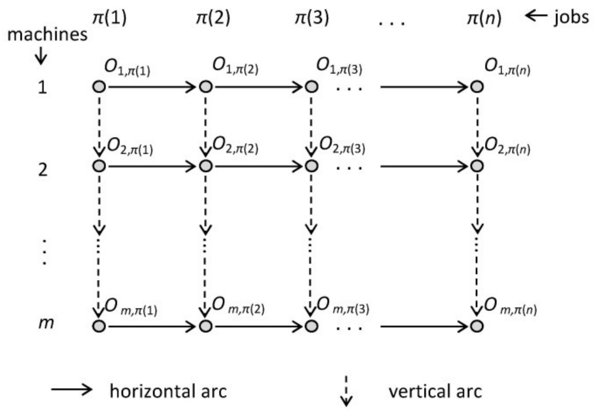

Fig. 1. Solution graph π

A sequence of vertices $(v_1,v_2,\ldots,v_w)$ of graph $G(\pi)$ such that $(v_i,v_{i+1})\in A(\pi)$, $i=1,2,...,w-1$ call path from vertex v_1 to v_w. Its length is equal to the sum of the weights of vertices and arcs (including the weight of the first and last vertex). By $P(u,v)$, $u,v\in O$ we denoted the critical path, i.e. the longest path in the graph from vertex u to v, and by $L(u,v)$ – the length of the path.

Example 1. Figure 2 shows a fragment of a permutation graph $\pi=(1,2,3,4,5,6,7)$ for data taken from the Table 1. A sequence of vertices $(O_{1,1},O_{1,2},O_{1,3},O_{1,4},O_{2,4},O_{3,4})$ is a path from vertex $O_{1,1}$ to $O_{3,4}$ of length 33. A critical path is $P(O_{1,1},O_{3,4})=(O_{1,1},O_{2,1},O_{3,1},O_{3,2},O_{3,3},O_{3,4})$, and its length $L(O_{1,1},O_{3,4})=47$.

Remark 2. The permutation $\pi\in\Phi$ is a feasible solution of the **CPwO** problem if and only if the graph $G(\pi)$ does not contain cycles.

Remark 3. Time of jobs execution, according to the solution $\pi\in\Phi$ equals to the length of the critical path (i.e. the longest path) of the graph $G(\pi)$.

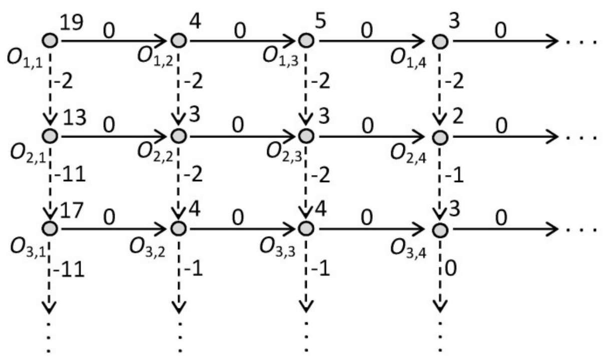

Fig. 2. Fragment of a solution graph for data from Example 1

Table 1. Total times of actions on working segments and overlaps values represented as workdays

works	*Segments*						
	***s*1**	***s*2**	***s*3**	***s*4**	***s*5**	***s*6**	***s*7**
***w*1**	19 (0)*	4 (0)	5 (0)	3 (0)	4 (0)	12 (0)	6 (0)
***w*2**	13 (2)	3 (2)	3 (2)	2 (2)	4 (2)	9 (2)	6 (2)
***w*3**	17 (11)	4 (2)	4 (2)	3 (1)	6 (2)	11 (7)	8 (4)
***w*4**	13 (11)	3 (1)	3 (1)	2 (0)	4 (2)	9 (7)	6 (4)
***w*5**	7 (5)	2 (0)	2 (0)	2 (0)	2 (0)	5 (3)	2 (0)
***w*6**	15 (13)	3 (3)	3 (2)	2 (1)	4 (4)	19 (9)	6 (4)
***w*7**	15 (15)	2 (3)	4 (3)	3 (2)	6 (4)	10 (10)	6 (6)
***w*8**	15 (15)	4 (2)	4 (4)	3 (3)	6 (6)	11 (10)	10 (6)

*in parentheses – overlap values.

Remark 4. $P(O_{1,\pi(1)},O_{m,\pi(n)})$ is a critical path in the graph $G(\pi)$.

Solving the **CPwO** problem boils down to determining such a feasible solution $\pi\in\Phi$ for which the corresponding graph $G(\pi)$ has the shortest possible critical path.

Let $\pi\in\Phi$ be a feasible solution. By $P(\pi)=(O_{1,\pi(1)},...,O_{m,\pi(n)})$ denote the critical path in the graph $G(\pi)$. This path can be partitioned into subsequences of vertices (subsequences of operations):

$$B=[B^1,B^2,\ldots,B^m],$$

such as:

(a) each subsequence contains successive operations executed directly one after another on the same machine;

(b) cross-section of any two subsequences is an empty set;

(c) each subsequence is a maximum (due to inclusion) the subset of the operations.

Subsequence B^k, $k=1,2,\ldots,m$ of operations from the critical path executed on the k-th machine is called a **block** (Wodecki 2009).

Remark 5. Any block contains exactly one operation from each task.

In the further part of the paper we will be considering blocks as sequences of jobs which will be denoted as follows:

$B^k = (\pi(a^k), \pi(a^k+1), \dots, \pi(b^k-1), \pi(b^k)),\ 1 \le a^k \le b^k \le n.$

Jobs $\pi(a^k)$ and $\pi(b^k)$ are respectively *the first* and *the last* in a sequence. In turn a block without the first and the last job ($\hat{B}^k$) we call an *internal* block. The definitions given are presented on the Figure 3.

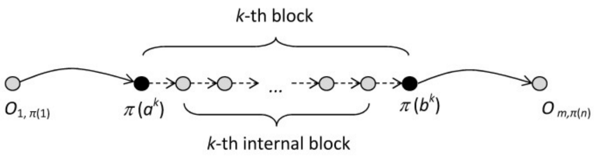

Fig. 3. A block on the critical path

Theorem 1. Let permutation $\pi \in \Phi$ be a feasible solution of construction project problem with overlaps. If B^k is a block from the critical path, then any change in the sequence of jobs of the internal block does not generate solution with cost function value lower than the value of the cost function of the solution π.

Proof. Let $B = [B^1, B^2, \dots, B^m]$ be a sequence of blocks in a permutation $\pi \in \Phi$. We are considering the *k*-th block:

$$B^k = (\pi(a^k), \pi(a^k+1), \dots, \pi(b^k-1), \pi(b^k)).$$

A critical path in a graph $G(\pi)$, i.e. from vertex $O_{1,\pi(1)}$ to $O_{m,\pi(n)}$:

$$P(\pi) = (B^1, B^2, \dots, B^m) = (B^1, B^2, ..., B^{k-1}, (O_{k,\pi(a^k)}, O_{k,\pi(a^k+1)}, \dots, O_{k,\pi(b^k-1)}, O_{k,\pi(b^k)}), B^{k+1}, ..., B^m) =$$
$$(P(O_{1,\pi(1)}, O_{k,\pi(a^k)}), P(O_{k,\pi(a^k+1)}, O_{k,\pi(b^k-1)}), P(O_{k,\pi(b^k)}, O_{m,\pi(n)}),$$

and its length:

$$L(\pi) = L(O_{1,\pi(1)}, O_{k,\pi(a^k)}) + L(O_{k,\pi(a^k+1)}, O_{k,\pi(b^k-1)}) + L(O_{k,\pi(b^k)}, O_{m,\pi(n)}) =$$
$$L(O_{1,\pi(1)}, O_{k,\pi(a^k)}) + \sum_{i=a^k+1}^{b^k-1} p_{k,\pi(i)} + L(O_{k,\pi(b^k)}, O_{m,\pi(n)}).$$

Let $\beta \in \Phi$ be a permutation generated from π by changing the order of the tasks of the internal block $\hat{B}^k$, i.e. tasks from a set of tasks:

$$\{\pi(a^k+1), \pi(a^k+2), \dots, \pi(b^k-2), \pi(b^k-1)\}.$$

From definition of a permutation β results that:

$$\beta(i) = \pi(i),\ i = 1, 2, ..., a^{k-1}, a^k, b^k, b^{k+1}, ..., n$$

and there appears equality of sets:

$$\{\beta(a^k+1), \beta(a^k+2), \dots, \beta(b^k-2), \beta(b^k-1)\} = \{\pi(a^k+1), \pi(a^k+2), \dots, \pi(b^k-2), \pi(b^k-1)\}.$$

In the graph $G(\beta)$ we are considering certain path from vertex $O_{1,\beta(1)}$ to $O_{m,\beta(n)}$:

$$W = (P(O_{1,\beta(1)}, O_{k,\beta(a^k)}), P(O_{k,\beta(a^k+1)}, O_{k,\beta(b^k-1)}), P(O_{k,\beta(b^k)}, O_{m,\beta(n)})).$$

The length of the path is:

$$L(W) = L(O_{1,\beta(1)}, O_{k,\beta(a^k)}) + L(O_{k,\beta(a^k+1)}, O_{k,\beta(b^k-1)}) + L(O_{k,\beta(b^k)}, O_{m,\beta(n)}).$$

From definition of a permutation β results that the length of some paths in graphs $G(\beta)$ and $G(\pi)$ are the same, in particular:

$$L(O_{1,\beta(1)}, O_{k,\beta(a^k)}) = L(O_{1,\pi(1)}, O_{k,\pi(a^k)});$$

$$L(O_{k,\beta(b^k)}, O_{m,\beta(n)}) = L(O_{k,\pi(b^k)}, O_{m,\pi(n)}).$$

Moreover:

$$L(O_{k,\beta(a^k+1)}, O_{k,\beta(b^k-1)}) = \sum_{i=a^k+1}^{b^k-1} p_{k,\beta(i)} = \sum_{i=a^k+1}^{b^k-1} p_{k,\pi(i)} = L(O_{k,\pi(a^k+1)}, O_{k,\pi(b^k-1)}).$$

Therefore, the length of the path W, $L(W) = L(\pi) = C_{\max}(\pi)$. The length of the critical path in the graph $G(\beta)$, $C_{\max}(\beta) \ge L(W)$ (because the critical path is the longest path in the graph). Thus $C_{\max}(\beta) \ge L(W) = C_{\max}(\pi)$.

We have shown that there exists a path W in the graph $G(\beta)$ which length equals:

$$L(W) = C_{\max}(\pi). \quad (9)$$

Because $C_{\max}(\beta)$ is the length of the longest path in the graph $G(\beta)$, therefore:

$$C_{\max}(\beta) \ge L(W). \quad (10)$$

From Eqns (9) and (10) we have $C_{\max}(\beta) \ge C_{\max}(\alpha)$.

In this way we have proved that if the permutation β was generated from π by changing the order of tasks in some internal block, then the length of the critical path $C_{\max}(\beta) \ge C_{\max}(\pi)$, which completes the proof.

The change of operations order in any block does not generate the solution with lower value of the cost function. At least one operation from any block should be moved before the first or after the last operation of this block to generate the solution (graph) with smaller length of the critical path. We use this property to reduce the neighborhood size, i.e. to omit generating solutions with greater values (comparing to the current solution) of the cost function.

3. Solution algorithms

In the literature several criteria of classification methods for solving *NP-hard* problems are used. With reference to the determined solutions the following methods are distinguished:

- exact methods;
- approximation methods.

Exact (optimal) methods, due to their exponential computation time, are of little use in practice. With their use, it is possible to solve only small size instances within a reasonable time. Even constantly increasing power of computers cannot significantly affect the increase in the size of examples solvable in acceptable time. Therefore, to solve *NP-hard* optimization problems there are almost exclusively approximate methods used. However, they do not give a guarantee that the determined solution is optimal. In reference to the way in which the solutions are determined algorithms can be divided into three main classes:

- construction algorithms;
- correction algorithms;
- metaheuristc algorithm.

Hereby considered **CPwO** problem is an important generalization of the flow shop problem. In recent years, many metaheuristic algorithms, solving the above mentioned flow shop problem, have been published. The best of them are based on tabu search method parallel algorithm (Grabowski, Wodecki 2004; Bożejko, Wodecki 2002). Solutions determined by them, for a representative group of examples, are only slightly different from the best currently known in the literature. Therefore, in order to solve the **CPwO**problem we will use the tabu search algorithm.

Construction algorithm. To determine the starting solution, for the tabu search algorithm, we have used a modified version of the NEH algorithm (Nawaz *et al.* 1983). This algorithm is considered to be the best algorithm for solving a classical design flow shop problem with the criterion $C_{\max}$. Based on computational experiments carried out on a representative group of examples, the mean relative error set by the NEH algorithm of solutions does not exceed 5%.

Algoritm MNEH

```
Step 1: Number the tasks so that:
    $t_1 \geq t_2 \geq, \ldots, \geq t_n$, where $t_j = \sum_{i=1}^{m}(p_{i,j} - l_{i,j})$;
    $\pi(1) := 1$;
Step 2: for $k := 2$ to $n$ to
    insert task $k$ in one of the positions $1, 2, \ldots, k$ in
    subpermutation $\pi$ so that the value of the objective
    function (i.e. the date of tasks completion) was
    minimal.
```

Computational complexity of the algorithm equals $O(m + n\ln n)$.

Tabu search method. This method was proposed by Glover and Laguna (1997) and then developed by other authors (Nowicki, Smutnicki 1996; Grabowski, Wodecki 2004). Tabu search (TS) is a modification of the local search method. In the basic version the TS method starts its operation of a certain initial solution $x^0 \in X$, where X is a set of the all feasible solutions. In the basic step of this method the whole neighbourhood $N(x^i)$ of a solution x^i is searched. Neighbourhood is defined by movements which can be performed with x^i. The search aims at finding in $N(x^{i+1})$ a solution with the smallest cost function value. The process of searching is continued till the best found solution.

To prevent the cyclic repetitions of solutions, stopping in a local extreme and to direct search into promising solutions areas there has been introduced the history memory of searching in the form of tabu list (so-called tabu list). On this list the definite number of recently "visited" solutions (basic solutions) is stored. These solutions are not remembered directly, but as some of their "attributes". That's why tabu arising from a particular iteration refers also to solutions which have been previously basic solutions. To weaken this constraint, additionally some aspiration function for the tabu solution is defined. If value of this function is smaller than the given level, then the solution is not treated as tabu. In the course of performing next iterations, it's remembered the best found solution x^* in the sense of cost function value and corresponding to the value of this function. Searching stops at the moment of starting working appropriate alloy's conditions. Summing up, the tabu method includes the following basic elements:

initial solution	–	a solution from which an algorithm starts its operation;
movement	–	a function transforming one solution into another;
neighbourhood	–	a set of solutions which are possible to obtain from a determined solution by means of a class of movements;
tabu list	–	a list including attributes of movements or solutions for determined number of recently examined solutions;
completion condition	–	a situation in which an algorithm ends up its operation e.g. (1) was performed the determined beforehand number of iterations; passed the algorithm's operation time, (2) in successive iterations the cost function value was constant.

Moves and neighbourhood. In the literature we can meet many types of a move based on interchanges of jobs on a machine. The intuition following from Theorem 1 suggests that the "insertion" type move should be the most proper one for the problem considered. Roughly speaking, the insertion move operates (see Wodecki 2009; Bożejko, Wodecki 2007) on a sequence of jobs on a machine and removes a job placed at a position in this sequence and inserts it in another position of the sequence. More precisely, let B^k, $k = 1, 2, \ldots, m$ be the kth block in a permutation $\pi \in \Phi$. For the job

$\pi(j) \in \hat{B}^k \cup \{\pi(b^k)\}$ let us denote by $\overleftarrow{N}^k(j)$ a set of permutations created by moving the job $\pi(j)$ to the beginning of the block B^k (before the first job in block $\pi(a^k)$). Analogously, for the $\pi(j) \in \hat{B}^k \cup \{\pi(a^k)\}$ let us denote by $\overrightarrow{N}^k(j)$ a set of permutations created by moving the job $\pi(j)$ to the end of the block B^k (after the last job in the block $\pi(b^k)$). The *neighbourhood* of the solution π has the form of:

$$N(\pi) = \bigcup_{k=1}^{m} \bigcup_{\pi(j) \in B^k} (\overleftarrow{N}^k(j) \cup \overrightarrow{N}^k(j)). \tag{11}$$

The number of elements of this neighbourhood is limited in advance by *n*.

Tabu List. Let π_l^k be the permutation obtained from π by removing job the $\pi(k)$ from its original position k, and next inserting it in the position l in π (i_l^k move). To prevent generating a cycle in the graph $G(\pi)$ some attributes of each movement are put on the list of tabu moves, which is served as the FIFO queue. We put the move's attribute, the triple $(\pi(k), l, F(\pi_l^k))$, on the tabu list L_{TS}.

Let us assume that we analyze a move i_l^k which generates the permutation β_l^k from $\beta \in \Phi$. If the triple (r, j, Ψ) such that $\beta(k) = r$, $l = j$ and $F(\beta_l^k) \geq \Psi$ is on the L_{TS} list, such a move is forbidden. The only parameter of this list is its length (the number of its elements). There are many realizations of the tabu list presented in the given references (Glover, Laguna 1997; Grabowski, Wodecki 2004).

The algorithm stops (*completion condition*) after *Max_iter*iterations.

Algorithm *ATS* (Tabu search)

```
Step 0: Find the initial solution $\pi_0$;
        $\pi := \pi_0$;
        $\pi^* := \pi$;
        $max\_tabu\_lenght = n/2$ ($n$ is the number of jobs);
Step 1: Define the neighbourhood $N(\pi)$ (see Eqn (11)) of a permutation $\pi$, remove from
        the neighbourhood elements forbidden by the tabu list except these $\beta \in N(\pi)$ for which:
        $C_{\max}(\beta) < C_{\max}(\pi)$;
Step 2: Find such a permutation $\delta \in N(\pi)$ for which:
        $C_{\max}(\delta) = \min\{C_{\max}(\beta) : \beta \in N(\pi)\}$;
Step 3: if $C_{\max}(\delta) < C_{\max}(\pi^*)$ then
        $\pi^* := \delta$;
        Insert attributes of $\delta$ to the list $L_{TS}$;
        if length ($L_{TS} > max\_tabu\_lenght$) then
            erase the oldest element on $L_{TS}$;
        $\pi := \delta$;
Step 4: if the completion condition is satisfied then
            Stop
        else go to Step 1.
```

4. Case study

During planning the road works execution, e.g. a road repair, the whole project can be divided into working parcels with different sizes whose boundaries are set for instance by crossroads/intersections with existing road paths. The sequence of taking actions on parcels by working teams will affect not only the total time of the whole project, but also the time of machines or working brigades delays. The problem of setting the optimal sequence of doing works on individual parcels in regard to the established criterion, for example the minimal time of carrying out a project, the minimal time of the working brigades delay or working costs concerning jobs sequencing.

To make a right division of works, it's necessary to determine the kind of works according to the general classification. The general works classification in the road and bridge construction is presented as follows:

1) preparatory works;
2) earthworks;
3) lands and pavements consolidation;
4) profiled lands and gravel pavements building;
5) reinforcement of soil-surfaced road pavement and subgrade;
6) broken stone pavement building;
7) asphalt concrete pavement building;
8) cement concrete pavement building;
9) repairing, conservation, maintenance and reconstruction of pavement;
10) production, conversion and purification of aggregates including aggregates usage obtained in the process of recycling;
11) loading, unloading and transport works;
12) bridges and culverts building;
13) energy production and transfer;
14) old objects demolition.

Example 1. We are considering a project for the construction of a section of a road, which was divided into seven segments: ***s*1, *s*2, *s*3, *s*4, *s*5, *s*6, *s*7**. Figure 4 shows a diagram of the works to be performed on each of the segments.

The technological order of works is as follows:

***w*1** – earthworks;
***w*2** – sand drainage blanket preparation;
***w*3** – preparation of broken-stone or macadam foundation;
***w*4** – binding course preparation with asphalt medium-grained concrete;
***w*5** – surface course preparation with asphalt fine-grained concrete;
***w*6** – roadsides preparation with stone dust;
***w*7** – drainage process;
***w*8** – planting and eventual topsoil removal.

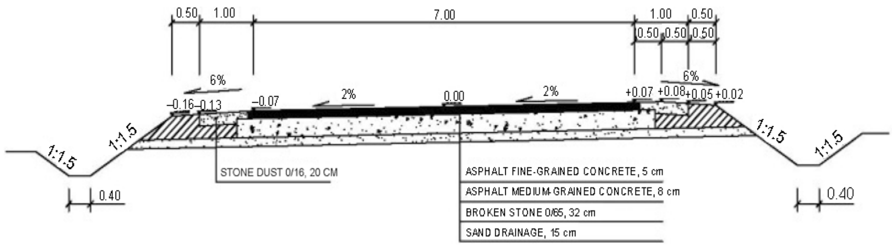

Fig. 4. The section of an access road to a dumping ground

For each of the segments the works should be done in a flow-shop system, i.e. in the following order: (***w*1, *w*2, *w*3, *w*4, *w*5, *w*6, *w*7, *w*8**). It is possible to start the implementation of the next job before the end of the previous one (overlap). Figure 5 shows one of the segments under construction. Some of the works are done in advance (in parallel). Figures illustrating the issue of a construction project example are given in Table 1.

Fig. 5. Works performed in advance (photo J. Chrobok)

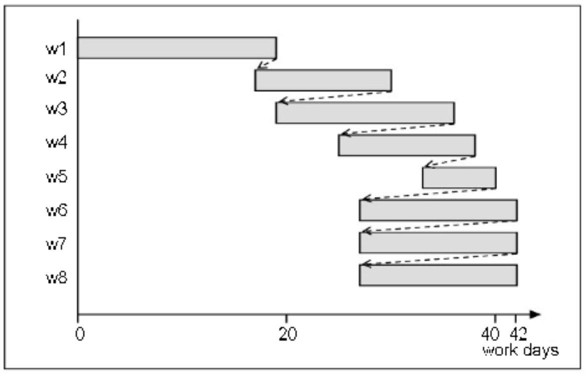

Fig. 6. Scheduling example for a road segment ***s*1**

We are presenting a special case of general works classification mentioned above, namely, the national road section, which consists of eight activities. Figure 6 shows works scheduling for a single road segment ***s*1** (the arrows indicate the order of execution of separate works). Total time of works completion took 42 workdays. The work ***w*1** is performed within 19 days. Two days before the end of the work **w1,*w*2** work begins (overlap for this work is 2, see Table 1). If the acceleration for each work from the segment ***s*1** was equal to 0 (as in the classic flow-shop system), then, in order to execute all the works in this segment, we would need 114 days.

4.1. Computational experiments

Due to the lack of test instances in the literature, for the considered problem appropriate test instances have been generated basing on instances of Taillard (1993) for a classical flow shop problem. Taillard's set of instances consists of 120 examples divided into 12 groups with different sizes. For each size $n\times m$: 20×5, 20×10, 20×20, 50×5, 50×10, 50×20, 100×5, 100×10, 100×20, 200×10, 200×20, 500×20 ten instances occur.

As in literature there are no solving algorithms for the problem considered, the quality of solutions (in the sense of the cost function value) generated by the ATS algorithm has been compared to solutions obtained by the MNEH algorithm. Additionally, there have been created instances of the problem of the small sizes nxm: 10×10. For these instances the optimal solution has been determined by a total searching of the solutions set (by the branch and bound, **B&B** algorithm). The obtained values have been compared to results which MNEH and ATS algorithms have provided.

To determine the initial permutation for the ATS algorithm the appropriately adopted MNEH algorithm has been applied. The adaptation of the this algorithm for the need of the algorithm discussed here has consisted in considering the transport times of jobs between machines during determining the cost function value for each solution processed in the algorithm.

For each problem instances the following values have been calculated:

C	–	the cost function value obtained by the ATS algorithm;
$PRD(A)$	–	the relative percentage deviation to the C^{ref} value obtained by the MNEH algorithm of the cost function value obtained by the ATSalgorithm; $PRD(A) = (C^A - C^{ref})/C^{ref} \cdot 100\%$.

Table 2 presents the percentage relative deviation to the MNEH algorithm for the tabu search algorithm considered. Table 3 shows the cost function value for the optimal solutions and solutions obtained by the MNEH and the tabu search algorithms. Average relative deviation of the constructive MNEH was on the level of 3.32%. The proposed tabu search found *optimal solutions* for all the examples (average relative deviation was 0.00%).

The proposed ATS algorithm as well as MNEH were also tested for the classic flow shop problem, i.e. for zero overlaps. The percentage relative deviation of the tabu search and the MNEH algorithms to the Taillard's (1993) reference solutions are presented in the Table 4. Average relative deviation to the best known solutions was on the level of 3.25% for the MNEH algorithm, 0.19% and 0.07% for the proposed tabu search algorithm, for 1000 and 5000 iterations, respectively.

Table 2. The percentage relative deviation for the tabu search algorithm *ATS* as against the MNEH algorithm

$n \times m$	*ATS* (1000 iter.)	*ATS* (2000 iter.)	*ATS* (5000 iter.)
20×5	–2.31%	–2.31%	–2.38%
20×10	–4.91%	–5.02%	–5.16%
20×20	–4.44%	–4.49%	–4.58%
50×5	–0.94%	–0.94%	–0.94%
50×10	–4.30%	–4.57%	–4.59%
50×20	–4.82%	–5.18%	–5.48%
100×5	–0.79%	–0.81%	–0.81%
average	–3.22%	–3.33%	–3.42%

Table 3. The cost function value for the optimal solution and solutions obtained by the MNEH and the tabu search algorithms

instance		C_{max}			*PRD*[%]	
	n×*m*	*OPT*	*MNEH*	*ATS*	*MNEH*	*ATS*
TM21	10×10	908	950	908	4.63	0.00
TM22	10×10	784	803	784	2.42	0.00
TM23	10×10	883	912	883	3.28	0.00
TM24	10×10	922	971	922	5.31	0.00
TM25	10×10	884	884	884	0.00	0.00
TM26	10×10	888	919	888	3.49	0.00
TM27	10×10	911	915	911	0.44	0.00
TM28	10×10	835	861	835	3.11	0.00
TM29	10×10	881	929	881	5.45	0.00
TM30	10×10	825	867	825	5.09	0.00
average					3.32	0.00

Table 4. The percentage relative effort of the tabu search and the MNEH algorithms as against Taillard's solutions

n×*m*	*MNEH*	*ATS* (1000 iter.)	*ATS* (5000 iter.)
20×5	3.36%	0.08%	0.02%
20×10	4.60%	0.27%	0.20%
20 20	3.73%	0.30%	0.22%
5 0 × 5	0.71%	0.16%	0.16%
50×10	4.14%	–0.02%	–0.12%
50×20	5.70%	0.52%	0.01%
100×5	0.50%	–0.01%	–0.01%
average	3.25%	0.19%	0.07%

Our algorithms MNEH and ATS was coded in C++ and executed on a PC with Intel Celeron 1.73 GHz processor. Computations times were at most few seconds.

From the perspective of practitioners of construction industry, there is an attempt to synchronize a number of works in time and space in scheduling of construction projects by flow-shop system. Computational experiments presented above gives practitioners an useful algorithms (ATS and fast MNEH) which can be embedded into expert systems and tool engineering computer applications, e.g. as it was shown in the work Bożejko *et al.* (2012).

4.2. Application of the proposed method to the case study

Apart of the benchmark data presented above, the practical data, referring to the building of road segments with different length, have been considered. In Table 1 here have been included data presented as the total times of actions on working segments represented as workdays. The ATS algorithm has been applied to data from Table 1.

The algorithm work has resulted in a solution $\pi = (4,2,5,7,6,1,3)$ for which the cost function value is $C_{\max} = 75$ (workdays). In the Figure 8 there is presented a building schedule for individual road segments for the obtained permutation π. The cost function value in case of scheduling for the permutation accordant to the segments numeration (natural permutation) $\pi = (1,2,3,4,5,6,7)$ (Fig. 7, $C_{\max} = 78$) is bigger than for a scheduling achieved from results of the ATS algorithm operation (Fig. 8, $C_{\max} = 75$). The algorithm executed 20 iterations and computations time did not exceed 0.1 sec.

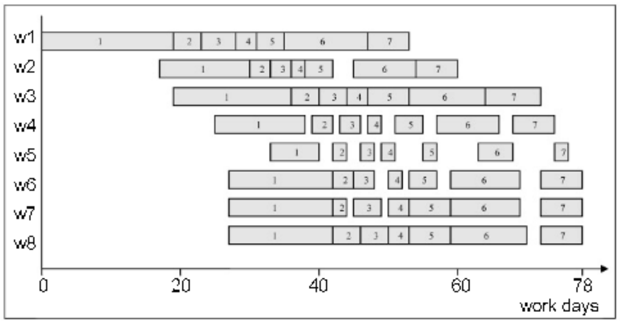

Fig. 7. Building schedule for individual road segments for the permutation accordant to the segments numeration (natural permutation)

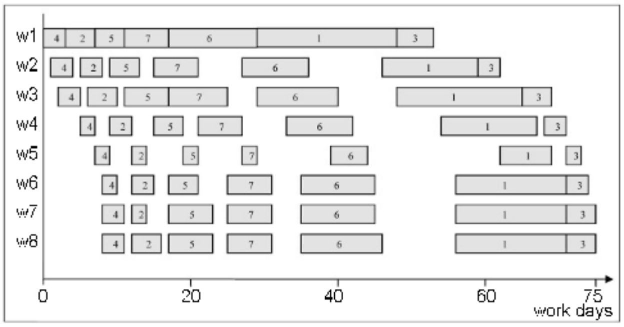

Fig. 8. Building schedule for individual road segments for the permutation obtained by the tabu search algorithm

In order to examine the effectiveness of the method there were additional calculations performed. They were carried out in two stages:

1. Using the algorithm of solving the flow shop problem (flow shop problem without acceleration of works) stated in the work of Grabowski and Wodecki (2004) the solution for the example taken from Table 1 was determined.

2. Given the acceleration (Table 1), moved up to the left (reduced) schedule of individual works was obtained.

   This resulted in a solution $C_{\max} = 98$ workdays.

Taking into account the acceleration of works already in the mathematical model of the issue provides a significant reduction (by over 20) of the completion time of the construction process.

Conclusions

We have discussed the problem of road projects scheduling in this paper. Moving the issues of construction scheduling in the field of classical scheduling theory one can encounter many difficulties associated with the selection of the correct model (a type of an issue), and the corresponding algorithm. For they are usually completely new, with many nonstandard constraints, strongly NP-hard combinatorial optimization problems. In order to solve them one can use the most up-to-date effective methods of algorithm construction (metaheuristics): simulated annealing, genetic algorithm, tabu search, neural networks, ant colony optimization, etc. On the basis of our theoretical research, experimental computations and the results presented in the literature (including "no free lunch" theorem) it can be stated that none of these methods is generally better than the others. Effective solving of NP-hard problems requires a proper choice of algorithm design methods and the use of specific properties of the problem.

We propose to use the new flow shop scheduling model with overlaps. Tabu search metaheuristics has been adopted to solve the problem with the use of the new so-called block properties which were introduced in the paper. Although the limitation of the proposed algorithms does not guarantee optimality of the generated solutions, however, solutions determined by the ATS algorithm are very close to optimal (for large examples of the 500 jobs the average percentage relative deviation to optimal ones equals 0.07%). The above mentioned algorithm is deterministic and guarantees repetition of calculations. Therefore, the obtained results show the efficiency of the proposed method, which is especially visible for data instances of the large size, as well as for the practical data of the small and medium sizes, for which the proposed tabu search approach gives optimal solutions.

Presented in the paper method allows its practitioners for a better analysis of the actual construction process and to determine the optimal (or close to optimal) schedules. This enables efficient use of not only equipment but also personnel and a better estimation of the time (and cost) of construction, which has a direct impact on the financial situation of construction companies.

Acknowledgements

The work was supported by the OPUS grant DEC-2012/05/B/ST7/00102 of Polish National Centre of Science.

References

Arditi, D.; Tokdemir, O. B.; Suh, K. 2001. Effect of learning on line-of-balance scheduling, *International Journal of Project Management* 19(5): 265–277. http://dx.doi.org/10.1016/S0263-7863(99)00079-4

Bożejko, W.; Wodecki, M. 2002. Solving the flow shop problem by parallel tabu search, in *Proc. of International Conference on Parallel Computing in Electrical Engineering (PARELEC '02)*, 2002, PR 01730: 189–194. http://dx.doi.org/10.1109/PCEE.2002.1115237

Bożejko, W.; Wodecki, M. 2004. Parallel genetic algorithm for the flow shop scheduling problem, in *Proc. of the 5th International Conference "Parallel Processing and Applied Mathematics" (PPAM 2003)*, 7–10 September 2003, Czestochowa, Poland. *Lecture Notes in Computer Science* 3019: 566–571.

Bożejko, W.; Wodecki, M. 2007. On the theoretical properties of swap multimoves, *Operations Research Letters* 35(2): 227–231. http://dx.doi.org/10.1016/j.orl.2006.03.002

Bożejko, W.; Wodecki, M. 2008. Parallel scatter search algorithm for the flow shop sequencing problem, in *Proc. of the 7th International Conference on Parallel Processing and Applied Mathematics (PPAM'07)*, 9–12 September 2007, Gdansk, Poland. *Lecture Notes in Computer Science* 4967: 180–188.

Bożejko, W.; Hejducki, Z.; Wodecki, M. 2012. Applying metaheuristic strategies in construction projects management, *Journal of Civil Engineering and Management* 18(5): 621–630. http://dx.doi.org/10.3846/13923730.2012.719837

Chrzanowski, Jr., E. N.; Johnston, D. W. 1996. Application of linear scheduling, *Journal of Construction Engineering and Management* 112(4): 476–491. http://dx.doi.org/10.1061/(ASCE)0733-9364(1986)112:4(476)

El-Rayes, K.; Moselhi, O. 2001. Optimizing resource utilization for repetitive construction projects, *Journal of Construction Engineering and Management* 127(1): 18–27. http://dx.doi.org/10.1061/(ASCE)0733-9364(2001)127:1(18)

Glover, F.; Laguna, M. 1997. *Tabu search*. Massachusetts, USA: Kluwer Academic Publishers. 382 p. http://dx.doi.org/10.1007/978-1-4615-6089-0

Grabowski, J.; Wodecki, M. 2004. A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion, *Computers & Operations Research* 31(11): 1891–1909. http://dx.doi.org/10.1016/S0305-0548(03)00145-X

Gupta, J.; Stafford, Jr., E. F. 2006. Flowshop scheduling research after five decades, *European Journal of Operational Research* 169(3): 699–711. http://dx.doi.org/10.1016/j.ejor.2005.02.001

Gupta, D. 2012. Branch and bound technique for three stage flow shop scheduling problem including breakdown interval and transportation time, *Journal of Information Engineering and Applications* 2(1): 24–29.

Hamerlink, D. J.; Rowings, J. E. 1998. Linear scheduling model: development of controlling activity path, *Journal of Construction Engineering and Management* 124(4): 263–268. http://dx.doi.org/10.1061/(ASCE)0733-9364(1998)124:4(263)

Harris, R. B.; Ionnou, P. G. 1998. Scheduling projects with repeating activities, *Journal of Construction Engineering and Management* 124(4): 269–278. http://dx.doi.org/10.1061/(ASCE)0733-9364(1998)124:4(269)

Hegazy, T. 1999. Optimization of construction time – cost trade-off analyses using genetic algorithms, *Canadian Journal of Civil Engineering* 26(6): 685–697. http://dx.doi.org/10.1139/l99-031

Ishubuchi, M.; Masaki, S.; Tanaka, H. 1995. Modified simulated annealing for the flow shop sequencing problems, *European Journal of Operational Research* 81(2): 388–398. http://dx.doi.org/10.1016/0377-2217(93)E0235-P

Johnston, D. W. 1984. Linear scheduling method for highway construction, *Journal of Construction Division* 107(2): 247–261.

Karwat, B. 2012. Optimization of production schedules in the steel production system, *Archives of Civil and Mechanical Engineering* 12(2): 240–252.

Lamoudan, T.; El Khoukhi, F.; El HilailiAlaoui, A.; Boukachour, J. 2011. Flow shop scheduling problem with transportation times, two-robots and inputs/outputs with limited capacity, *International Journal of Intelligent Computing Research* 2(1/2/3/4): 244–253.

Lucko, G.; PeñaOrozco, A. A. 2009. Float types in linear schedule analysis with singularity functions, *Journal of Construction Engineering and Management* 135(5): 368–377. http://dx.doi.org/10.1061/(ASCE)CO.1943-7862.0000007

Mattila, K.; Abraham, D. 1998. Resource leveling of linear schedules using integer linear programming, *Journal of Construction Engineering and Management* 124(3): 232–244. http://dx.doi.org/10.1061/(ASCE)0733-9364(1998)124:3(232)

Mattila, K. G.; Park, A. 2003. Comparison of linear scheduling model and repetitive scheduling method, *Journal of Construction Engineering and Management* 129(1): 56–64. http://dx.doi.org/10.1061/(ASCE)0733-9364(2003)129:1(56)

Moselhi, O.; Lorterapong, P. 1995. Fuzzy vs probabilistic scheduling, in *Proc. of the 12th Conference "Automation and Robotics in Construction" (ISARC)*, 1995, Warsaw, Poland, 441–448.

Nawaz, M.; Enscore, E. E.; Ham, I. 1983. A heuristic algorithm for the *m*-machine, *n*-job flow-shop sequencing problem, *OMEGA* 11(1): 91–95. http://dx.doi.org/10.1016/0305-0483(83)90088-9

Nowicki, E.; Smutnicki, C. 1996. A fast tabu search algorithm for the permutation flow-shop problem, *European Journal of Operational Research* 91(1): 160–175. http://dx.doi.org/10.1016/0377-2217(95)00037-2

O'Brien, J. J. 1969. *Scheduling handbook*. New York: McGraw Hill. 605 p.

Reeves, C. R.; Yamada, T. 1998. Genetic algorithms, path relinking and the flowshop sequencing problem, *Evolutionary Computation* 6(1): 45–60. http://dx.doi.org/10.1162/evco.1998.6.1.45

Rogalska, M.; Bożejko, W.; Hejducki, Z. 2008. Time/cost optimization using hybrid evolutionary algorithm in construction project scheduling, *Automation in Construction* 18(1): 24–31. http://dx.doi.org/10.1016/j.autcon.2008.04.002

Taillard, E. 1993. Benchmarks for basic scheduling problems, *European Journal of Operational Research* 64(22): 278–285. http://dx.doi.org/10.1016/0377-2217(93)90182-M

Wodecki, M.; Bożejko, W. 2002. Solving the flow shop by parallel simulated annealing, in *Proc. of the 4th International Conference on Parallel Processing and Applied Mathematics (PPAM 2001)*, 9–12 September 2001, Nałęczów, Poland. *Lecture Notes in Computer Science* 2328: 236–244.

Wodecki, M. 2009. A block approach to earliness-tardiness scheduling problems, *International Journal on Advanced Manufacturing Technology* 40: 797–807. http://dx.doi.org/10.1007/s00170-008-1395-7

Wolpert, D. H.; Macready, W. G. 1997. No free lunch theorems for optimization, *IEEE Transactions on Evolutionary Computation* 1(1): 67–82. http://dx.doi.org/10.1109/4235.585893

Zavadskas, E. K.; Turskis, Z.; Tamošaitiene, J. 2010. Risk assessment of construction project, *Journal of Civil Engineering and Management* 16(1): 33–46.

Wojciech BOŻEJKO. Associated Professor at Wroclaw University of Technology. He obtained PhD in Wroclaw University of Technology in 2003, DSc (habilitation) in Wrocław University of Technology, Faculty of Electronics, in 2011 and Associated Professor position in Wrocław University of Technology in 2013. Author of over 150 papers in journals and conference proceedings from the field of parallel processing, scheduling and optimization. He is also a qualified musician, graduated (MA) from the Academy of Music in Wroclaw in a specialization of piano. He is interested in parallel algorithms, discrete optimization, scheduling and supercomputing.

Zdzisław HEJDUCKI. Professor at the Institute of Building Engineering of Wroclaw University of Technology, Poland. Research interests: civil engineering and management, special scheduling methods and the optimization of discrete processes occurring directly in construction, focusing mainly on the application of artificial intelligence methods (simulating annealing, genetic algorithm, tabu search, neural networks).

Mariusz UCHRONSKI. PhD student at Wroclaw University of Technology. He received the MS degree in Electronics and Telecomunicstions with Computer Applied Engineering specialization from Wroclaw University of Technology in 2008. Co-author of several papers in journals and conference proceedings from the field of parallel processing, scheduling and optimization. His research interests include parallel programming, parallel algorithms design, discrete optimization, scheduling and HPC.

Mieczysław WODECKI. Is an Assistant Professor at Institute of Computer Science University of Wrocław. He obtained PhD in Wroclaw University of Technology, Institute of Computer Engineering, Control and Robotics in 1988 and DSc (habilitation) in Wrocław University of Technology, Faculty of Electronics, in 2011. He is an author of over 140 papers (and over 45 citations on ISI Web of Knowledge) in journals and conference proceedings from the field of parallel processing, scheduling and optimization. He is interested in discrete optimization and parallel algorithms.