Working Paper
n°: 2017-100-01

Optimal workforce assignment to operations of a paced assembly line

Alexandre Dolgui [a], Sergey Kovalev [b], Mikhail Y. Kovalyov [c], Sergey Malyutin [a], Ameur Soukhal [d]

a IMT Atlantique, LS2N, CNRS, La Chantrerie, 4 rue Alfred Kastler – B.P. 20722, F-44307 Nantes Cedex 3, France
b INSEEC Business School, 25 rue de l'Université, 69007 Lyon, France
c United Institute of Informatics Problems, National Academy of Sciences of Belarus, Minsk, Belarus
d Laboratoire d'Informatique, Ecole Polytechnique de l'Université de Tours, Tours, France

January 2017

An ulterior version of this article is to appear in European Journal of Operational Research.
It can be purchased at: https://doi.org/10.1016/j.ejor.2017.06.017

Optimal workforce assignment to operations of a paced assembly line

Alexander Dolgui[1], Sergey Kovalev[2], Mikhail Y. Kovalyov[3], Sergey Malyutin[1], Ameur Soukhal[4]

[1]*Automation, Production and Computer Sciences Dept., Ecole des Mines de Nantes, IRCCYN, UMR CNRS 6597, La Chantrerie 4, rue Alfred Kastler - B.P. 20722, F-44307 Nantes Cedex 3, France, E-mail: alexandre.dolgui@mines-nantes.fr*

[2]*INSEEC Business School, 25 rue de l'Université, 69007 Lyon, France, E-mail: skovalev@inseec.com*

[3]*United Institute of Informatics Problems, National Academy of Sciences of Belarus, Minsk, Belarus, E-mail: kovalyov_my@newman.bas-net.by*

[4]*Laboratoire d'Informatique, Ecole Polytechnique de l'Université de Tours,Tours, France, E-mail: ameur.soukhal@univ-tours.fr*

Abstract

We study a paced assembly line intended for manufacturing different products. Workers with identical skills perform non-preemptable operations, whose assignement to stations is known. Operations assigned to the same station are executed sequentially, and they should follow the given precedence relations. Operations assigned to different stations can be performed in parallel. The operation's processing time depends on the number of workers performing this operation. The problem consists in assigning workers to operations such that the maximal number of workers employed simultaneously in the assembly line is minimized, the line cycle time is not exceeded and the box constraints specifying the possible number of workers for each operation are not violated. We show that the general problem is NP-hard in the strong sense, develop conventional and randomized heuristics, propose a reduction to a series of feasibility problems, present a MILP model for the feasibility problem, show relation of the feasibility problem to multi-mode project scheduling and multiprocessor scheduling, establish computational complexity of several special cases based on this relation and provide computer experiments with real and simulated data.

Keywords: workforce assignment, production line, optimization, mixed integer linear programming, heuristics

1 Introduction

We study a *paced unidirectional assembly line* consisting of m stations and manufacturing different products. Every station switches from processing a current product to the next one simultaneously. The time interval between two consecutive switches is called *cycle* and its duration is called *cycle time*. Cycle time remains the same for every cycle. Motivated by an industrial case, we study a workforce assignment problem for a single cycle of such line. Without loss of generality, it is assumed that the cycle starts at time zero. In a given cycle, workers at station k execute a given set N_k of operations, $k = 1, \dots, m$. Parallel execution of operations is possible if these operations belong to different stations. On the other hand, the order of operations assigned to the same station should follow a given technological process characterized by *precedence relations* between the operations. If operation i is followed by operation j, then i must be completed before the start time of j. If operations i and j have no precedence relation, then they are called *independent* and can be performed in any order. Operations without predecessors can start at time zero. The set of precedence relations of operations at station k is represented by a *directed acyclic graph* $G_k = (N_k, U_k)$, where N_k is the set of operations assigned to station k and U_k, $U_k \subset N_k \times N_k$, is the set of oriented pairs of operations (i, j) of station k such that $(i, j) \in U_k$ if and only if operation i is followed by operation j. Let $N = \cup_{k=1}^{m} N_k$, $n = |N|$, and $U = \cup_{k=1}^{m} U_k$. Define graph $G = (N, U)$.

Operations are executed by at most $r_{\max}$ identical workers. *Processing time* $p_j(r)$ of an operation j is a positive non-increasing function of the number of workers r assigned to this operation. Operations are non-preemptive, and if a worker starts performing an operation, he or she cannot switch to any other operation before finishing the current one. Workers cannot execute more than one operation simultaneously. The time spent by a worker to move from one station to another is negligibly small compared to any processing time of any operation. Therefore, it is assumed that any worker can move from one operation to another in zero time. Workforce assignment consists in creating a *schedule*, in which the start time of each operation, its processing time and the sequence of operations for each worker are determined. Given a schedule, the number r_j of workers assigned to operation j, the operation *start time* S_j and the operation *completion time* C_j can be calculated for each operation j such that $C_j = S_j + p_j(r_j)$, $j = 1, \dots, n$. The *makespan* of a schedule is defined as $C_{\max} = \max_{j \in N}\{C_j\}$. This value is equal to the line cycle time.

The following constraints must be satisfied in a *feasible* schedule:

- *Box constraints.* For technical reasons, the number of workers assigned to an operation must be within certain limits: $a_j \le r_j \le b_j$, where a_j and b_j are given positive integer numbers, $j = 1, \ldots, n$.

- *Cycle time constraint.* In order to achieve the desired level of productivity, the line cycle time must not be exceeded: $C_{\max} \le d$, $j = 1, \ldots, n$, where d is a given upper bound on the line cycle time.

The criterion of the problem, that we denote as $MinNumber$, is the minimization of the maximal number of workers employed simultaneously in the line,

$$W_{\max} = \max_{0 \le t \le d} \Big\{ \sum_{j \in N(t)} r_j \Big\},$$

where $N(t)$ is the set of operations executed at time instant t. Let $W^*_{\max}$ denote the minimal $W_{\max}$ value. Assume without loss of generality that the number of available workers is such that $r_{\max} \le \sum_{j \in N} b_j$, because otherwise we can reset $r_{\max} = \sum_{j \in N} b_j$, and that $\sum_{j \in N_k} p_j(b_j) \le d$ for $k = 1, \ldots, m$ and $\max\Big\{ \max_{j \in N}\{a_j\}, \Big\lceil \sum_{j \in N} p_j(b_j)/d \Big\rceil \Big\} \le r_{\max}$, because otherwise the problem $MinNumber$ has no solution.

For the sake of clarity, consider an example in which the assembly line consists of two stations. There are eight operations and four available workers. Precedence graph for this example is presented in Figure 1.

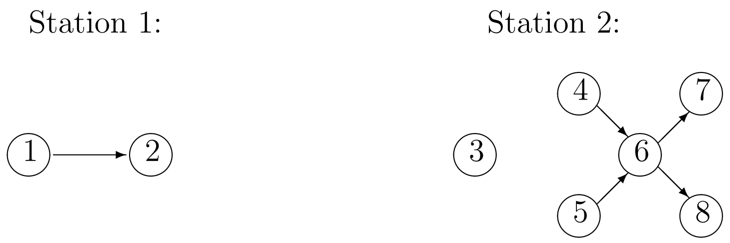


Figure 1: Precedence graph

Processing times of operations depending on the number of workers are given in Table 1.

An empty entry for a given number of workers and operation j means that this number of workers is either less than a_j or greater than b_j. Note that all four workers can be used on the line, but only one, two or three of them can be used to perform the same operation. A Gantt chart illustrating a feasible schedule with an upper bound on line cycle time $d = 44$ and maximum number of workers $r_{\max} = 4$ is given in Figure 2.

Operations\ Number of workers	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	50	10	10	9	11	24	20	
2	30		6	5	7	12	12	9
3						8	7	6

Table 1: Processing times of operations

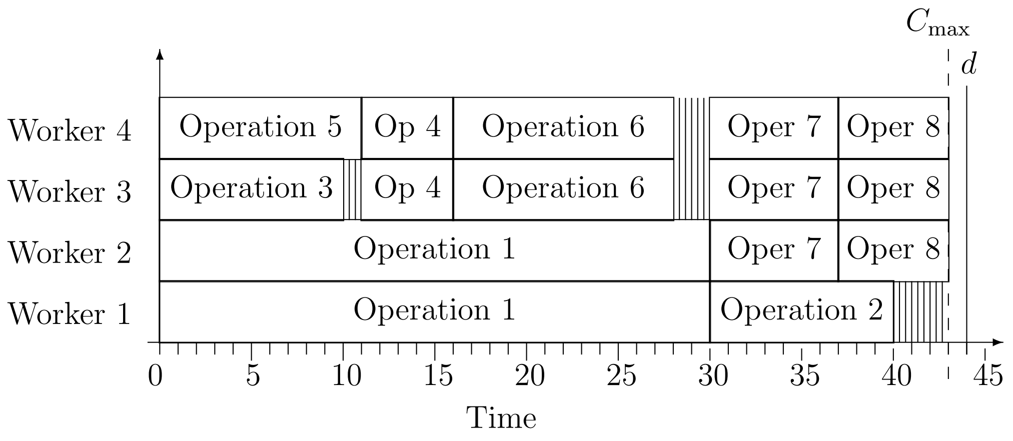


Figure 2: A feasible schedule. Dashed rectangles represent idle times of workers.

Problem *MinNumber* appeared as a subproblem in the European project amePLM [1] on workforce planning for an assembly line that produces three automobile engine models. Each engine model or its component visits stations in the same order and requires the same set of operations. Operation processing times depend on the engine model and they are inversely proportional to the number of used workers. A detailed description of this industrial case is given in Battaïa et al. [7]. Preliminary results of our studies of the problem *MinNumber* are presented at the 15th IFAC Symposium [22]. In [22], a special case of the problem *MinNumber* is addressed, in which graphs G_k are connected, as they are in the industrial case, though, this specificity is not precisely mentioned. The current paper extends, corrects and improves the draft heuristics and the MILP model in [22]. It additionally contains an updated literature review, a classification of computational complexity of special cases and a computational study. A particular result is that the number of workers in the real life problem of the project amePLM is decreased from 26, obtained by Battaïa et al. [7], to 25.

The rest of the paper is organized as follows. The next section presents a literature review. In Section 3, we present the exact bisection search procedure, which consists in an iterative solution of feasibility problems *Feasible*(Q) for a given number Q of workers. This section also provides the description of a *Mixed Integer Linear Programming (MILP)* model for problems *Feasible*(Q). In Section 4, we show the relationship of the problem *Feasible*(Q) to the multi-mode project scheduling problems and multiprocessor scheduling problems and establish the computational complexity of several special cases of the problem $MinNumber$ based on the relationship to the latter problems. Three constructive heuristics, two conventional ones and one randomized, are given in Section 5. Computational study of the heuristics and the MILP model is described in Section 6. Section 7 contains a summary of the results and suggestions for future research.

2 Literature review

The earliest studies of optimal workforce assignment problems concentrated on the two-dimensional assignment models with simple constraints. They were initiated in the 19th century and have become classics of combinatorial optimization, see the monograph of Burkard et al. [16]. Later on, a number of practical constraints were taken into consideration in timetabling, rostering, shift scheduling and resource constrained project scheduling models, as it was described, e.g., in Willemen [62], Ernst et al. [26], Naveh et al. [46], Rocha et al. [50], Miller [41] and Artigues et al. [5].

There exists a vast body of literature on the assignment of workers to operations of a production line. Vidic [58] studies the effect of the assignment of a fully cross-trained workforce on the throughput of a serial production line. She focuses on dynamic worksharing and fixed workforce assignment and suggests two models. One model assumes that workers performance is determined by their steady-state productivity rate and the other model assumes that workers' productivity rates depend of their learning and forgetting characteristics. Heuristic methods are developed. A recent review and classification of the literature related to workforce planning problems with skills is presented by De Bruecker et al. [20]. Corominas et al. [19] study a problem in which skilled permanent and unskilled temporary workers have to be assigned to an assembly line. Skilled workers require less time to finish a task than unskilled ones, and unskilled workers must work alongside at least one skilled worker. The criterion is to minimize the number of unskilled workers. The authors suggest a binary linear programming formulation for this problem. Blum and Miralles [13] employ beam search heuristics for the problem of minimizing the production line cycle time, provided that the number of workers is fixed. Karabak et al. [31] study a workforce assignment problem in which some workers are qualified to perform a task while others are not. Walking durations between different tasks are taken into consideration. The criterion is to minimize the number of workers. Due to the high complexity of the problem, the authors apply heuristics for instances with up to 30 operations. Araujo et al. [4] consider the Assembly Line Worker Assignment and Balancing Problem (ALWABP) with non-identical workers. They are interested in the insertion of slow or limited workers into the production process. Two cases of this problem are studied: parallel workstations (one worker is assigned to the same station) and collaborative workstations (several workers are assigned to the same station). Although the criterion is to minimize the cycle time, the proposed solution method is concerned with the inverse problem of minimizing the number of stations, subject to an upper bound on the cycle time. Different cycle time upper bounds are iteratively tested starting from a lower bound. If a feasible solution for the desired number of stations is not found, then the problem is solved again with a marginally increased cycle time. Borba and Ritt [14] propose a MIP model, a heuristic algorithm based on a beam search, and a task oriented branch-and-bound procedure for the same problem with the non-identical workers. The case of uncertain worker dependent processing times, expressed by intervals of possible values, is considered by Moreira and Costa [44]. The goal is to integrate workers with disabilities in an assembly line while minimizing the number of extra stations needed. An extension of the ALWABP to minimize the expected cycle time under uncertain worker availability is proposed by Ritt et al. [49]. The case of non-identical workers of this problem is studied by Moreira and Costa [43]. A job rotation schedule has to be developed. Two goals are considered: minimization of the sum of cycle times and maximization of the number of different operations each worker executes in a complete rotation period, considering all the sub-period schedules. A three-steps solution procedure is used by the authors. The first step builds single period schedules. In the second step a MIP formulation is applied to find a complete rotation plan. The third step improves the obtained solutions. Mutlu et al. [45] develop iterative genetic algorithm for ALWABP with the cycle time minimization criterion. Vilá and Pereira [59] propose an exact enumeration procedure for the same problem. The new lower bounds allowed the authors to improve results for a benchmark set of instances. Koltai and Tatay [34] propose a general framework to model skill requirements and skill conditions for assembly line balancing models.

A simultaneous minimization of the number of workers and sequence-dependent setups in mixed-model assembly lines is studied by Giard and Jeunet [29]. The goal is to avoid line stoppages with the minimal number of temporarily hired utility workers. They propose an exact method allowing to obtain optimal solutions in reasonable time for instances with up to 15 production items. A new assembly line problem with qualification requirements of operations and qualification levels of workers is introduced by Sungur and Yavuz [53]. In the hierarchical workforce structure, a lower qualified worker can be substituted by a higher qualified one (but not vice versa) with a certain cost. The objective is to minimize the total cost, while respecting a given cycle time. An ILP model is used to solve this problem. The part supply scheduling problem in line-integrated supermarkets with the workforce minimization criterion and no stock out constraint is studied by Boysen and Emde [15]. The authors use a heuristic decomposition approach to solve this problem and draw important conclusions for managers. A scheduling problem with the criterion of workforce minimization is treated by Camm et al. [17]. The authors deal with a paced line composed of identical parallel workstations. The number of workers needed at a workstation depends on the job processed and it is fixed. A two-stage approach is applied. The first stage is a MILP model that determines starting times. The second stage is a polynomial time procedure which assigns jobs to specific assembly workstations.

There is a stream of publications in which workforce requirements of the operations are assumed to be fixed, there is one operation for any product on each station of the transfer line, and the problem is to find a cyclic sequence of products such that the maximum number of workers needed at any time is minimized. The relevant results can be found in Akagi et al. [3], Wilson [63], Lutz and Davis [40], Lee and Vairaktarakis [37], Vairaktarakis and Winch [57], Kouvelis and Karabati [36], Vairaktarakis et al. [56], Vairaktarakis and Cai [55] and Kovalyov et al. [35].

3 Exact bisection search procedure

In order to solve problem $MinNumber$ we use an exact *bisection search* procedure, which consists in an iterative solution of feasibility problems. In a feasibility problem we try to find a feasible schedule for a given number Q of workers. We denote this problem as $Feasible(Q)$. Let LB and UB be lower bound and upper bound, respectively, on the value of Q. The general description of the bisection search procedure is as follows.

Step 1 Calculate LB and UB.

- If a feasible solution of problem $Feasible(LB)$ is found, then it is an optimal solution of the problem $MinNumber$, stop. Otherwise, solve the problem $Feasible(UB)$.
- If a feasible solution of $Feasible(UB)$ is found, then go to Step 2. Otherwise, solve the problem $Feasible(r_{\max})$.

- If no feasible solution of $Feasible(r_{\max})$ is found, then the problem $MinNumber$ has no feasible solution, stop. Otherwise, set $UB = r_{\max}$ and go to Step 2.

Step 2 Generic iteration of bisection search.

- If $LB + 1 = UB$, then an optimal solution for the problem $MinNumber$ is found. It is the feasible schedule obtained for the problem $Feasible(UB)$ in Step 1.
- If $LB + 2 \le UB$, then set $Q = \lceil \frac{UB+LB}{2} \rceil$ and solve the problem $Feasible(Q)$. If a feasible solution for the problem $Feasible(Q)$ is found, then reset $UB := Q$ and repeat Step 2. Otherwise, reset $LB := Q$ and repeat Step 2.

Once passed to Step 2, the bisection search procedure delivers an optimal solution in $O(\log_2(UB - LB))$ iterations. The problem $Feasible(Q)$ for a certain Q, $Q \in \{LB, LB+1, \dots, UB\}$ is solved within each iteration.

We now describe a MILP model for the problem $Feasible(Q)$. It is convenient to introduce the following notation:

- N^-: subset of vertices from N, which have no successor;
- I: set of vertex pairs (i, j) that are independent with respect to the precedence constraints, $I = \{(i, j) \mid i \in N, j \in N, i \neq j, (i, j) \notin U, (j, i) \notin U\}$;
- $I_k = \{(i, j) \mid (i, j) \in I, i \in N_k,\ j \in N_k\}$, $k = 1, \dots, m$;
- $I_{gk} = \{(i, j) \mid (i, j) \in I, i \in N_g,\ j \in N_k\}$, $g = 1, \dots, m$, $k = 1, \dots, m$, $g \neq k$.

Decision variables are the following:

- $x_{ir} \in \{0, 1\}$, $i \in N$, $r = 1, \dots, Q$: $x_{ir} = 1$ if worker r is assigned to operation i, and $x_{ir} = 0$, otherwise;
- $y_{ij} \in \{0, 1\}$, $(i, j) \in I$: $y_{ij} = 1$ if operation j starts after or on the completion of operation i, and $y_{ij} = 0$, otherwise. If $y_{ij} = 0$ it means that one of two possible situations occurs:
  - i starts after or on the completion of j,
  - i and j are partially performed in parallel. In a feasible schedule i and j must be, in such case, performed by different workers.

- $z_{ir} \in \{0,1\}$, $i \in N$, $r = 1,\dots,Q$: $z_{ir} = 1$ if r workers execute operation i, and $z_{ir} = 0$, otherwise;
- $S_{ir} \geq 0$, $i \in N$, $r = 1,\dots,Q$: start time of operation i if it is executed by worker r, and any non-negative value otherwise.

A Mixed Integer Linear Programming (MILP) formulation of the problem $Feasible(Q)$ is as follows.

Problem $Feasible(Q)$:

$$S_{ir} + \sum_{q=1}^{Q} p_i(q) z_{iq} \leq d,\ i \in N^-,\ r = 1,\dots,Q, \tag{1}$$

$$\sum_{i\in N}\sum_{q=1}^{Q} p_i(q) z_{iq} \leq Qd, \tag{2}$$

$$\sum_{r=1}^{Q} z_{ir} = 1, \quad i \in N, \tag{3}$$

$$\sum_{r=1}^{Q} x_{ir} = \sum_{r=1}^{Q} r z_{ir}, \quad i \in N, \tag{4}$$

$$S_{jh} - S_{iq} \geq \sum_{r=1}^{Q} p_i(r) z_{ir},\ (i,j) \in U,\ h,q = 1,\dots,Q, \tag{5}$$

$$S_{jh} - S_{iq} \geq \sum_{r=1}^{Q} p_i(r) z_{ir} - (d+1)(3 - y_{ij} - x_{ih} - x_{jq}),\ (i,j) \in I_k,\ k = 1,\dots,m,\ h,q = 1,\dots,Q, \tag{6}$$

$$S_{jh} - S_{ih} \geq \sum_{r=1}^{Q} p_i(r) z_{ir} - (d+1)(3 - y_{ij} - x_{ih} - x_{jh}),\ (i,j) \in I_{gk},\ g,k = 1,\dots,m,\ g \neq k,\ h = 1,\dots,Q, \tag{7}$$

$$S_{ih} - S_{iq} \leq (d+1)(2 - x_{ih} - x_{iq}), \quad i \in N,\ h = 1,\dots,Q,\ q = h+1,\dots,Q, \tag{8}$$

$$S_{iq} - S_{ih} \leq (d+1)(2 - x_{ih} - x_{iq}), \quad i \in N,\ h = 1,\dots,Q,\ q = h+1,\dots,Q, \tag{9}$$

$$y_{ij} + y_{ji} \leq 1, \quad (i,j) \in I, \tag{10}$$

$$x_{ir} + x_{jr} - 1 \leq y_{ij} + y_{ji},\ (i,j) \in I,\ r = 1,\dots,Q, \tag{11}$$

$$\sum_{r=1}^{Q} r z_{ir} \leq b_i, \quad i \in N, \tag{12}$$

$$a_i \leq \sum_{r=1}^{Q} r z_{ir}, \quad i \in N, \tag{13}$$

$$x_{ir}, y_{ij}, z_{ir} \in \{0,1\}, \quad i,j \in N,\ r = 1,\dots,Q, \tag{14}$$

$$S_{ir} \geq 0, \quad i \in N,\ r = 1,\dots,Q. \tag{15}$$

Constraints (1) address the cycle time d on each station by considering completion times of operations that have no successors. Constraints (2) guarantee that n small rectangles with dimensions (number of workers, operation time) fit the rectangle with dimensions (Q, d). Constraints (3) oblige every operation to be processed by only one number of workers. Constraints (4) verify that if some operation i has to be executed by a certain number of workers, for instance v, and, therefore, $Z_{iv} = 1$, then exactly v variables x_{ir} are equal to 1. Constraints (5) require that the starting times of operations i and j performed by any number of workers are at least $p_i(r)$ time units away from each other if i precedes j. Constraints (6) enforce worker h to start operation j after or at the time when worker q completes operation i, if i and j belong to the same station k. In such case $y_{ij} = 1$, $x_{iq} = 1$, $x_{jh} = 1$. Constraints (7) do the same as (6), but in the case when operations i and j are on different stations and they are performed by the same worker. Constraints (8) and (9) force all workers assigned to the same operation to start at the same time. Constraints (10) guarantee that, for a pair of independent operations i and j, i.e. operations without precedence relation between them, the situation in which both $y_{ij} = 1$ and $y_{ji} = 1$ does not occur. Constraints (11) ensure that if a pair of independent operations i and j are assigned to the same worker r, then one of them must start after or on the completion of the other, i.e., $y_{ij} = 1$ and $y_{ji} = 1$ cannot happen at the same time. Suppose, for example, that a pair of independent operations i and j are performed by the same worker r. In such case $x_{ir} = 1$ and $x_{jr} = 1$. Constraints (10) and (11) ensure that $y_{ij} = 1$ and $y_{ji} = 1$ cannot happen at the same time. Besides, constraints (5), for which $r = h = q$, rule out a simultaneous execution of operations i and j by worker r. If, for example, worker r is assigned to only one of two operations i and j, or not assigned to any of them, then operations i and j can be performed independently in time. In such case $y_{ij} = 0$ and $y_{ji} = 0$. Constraints (12) and (13) verify that the number of workers assigned to every operation respects the predetermined limits.

Building a schedule implies the knowledge of operations' start times, their durations and sequence for every worker. A solution the MILP problem $Feasible(Q)$ provides this information. The MILP model is applied in the computational study in Section 6.

4 Relation to multi-mode project scheduling and multiprocessor scheduling. Computational complexity

The problem $Feasible(Q)$ can be viewed as a multi-mode project scheduling problem, in which an activity (operation) is assigned a mode (set of workers) and has a mode-dependent duration, see Kolisch et al. [33], Tseng and Chen [54] and Artigues et al. [5] for the definitions of the latter problem. Mathematical programming formulations of multi-mode project scheduling problems encountered in literature have decision variables with indices whose number equals the number of modes, see Kolisch and Sprecher [32]. A solution of the problem $Feasible(Q)$ should specify not only the number of workers executing an operation, but also identify them. Therefore, there can be $2^Q - 1$ different modes and as many variables in a problem's formulation based on modes. In order to solve multi-mode project scheduling problems, authors often recur to branch-and-bound methods, metaheuristics, time- and event-indexed MILP formulations, see, for example, Monma et al. [42], Demeulemeester et al. [21], Salewski et al. [51], Ranjbar and Kianfar [48], Li and Womer [38], Besikci et al. [12], and Ghoddousi et al. [28].

Problem $Feasible(Q)$ is related to the multiprocessor *moldable* task scheduling problem, in which the number of identical processors allocated to a computer task influences its processing time. Once a task is started, the number of allocated processors cannot be changed. Processors play the role of workers. Box constraints on the number of processors allocated to the same task are not considered.

Drozdowski [23] provides a state-of-the-art of multiprocessor task scheduling problems. According to this paper, the makespan minimization equivalent of the problem $Feasible(Q)$, in which a single operation is assigned to each station, is denoted as $P|spdp - lin - \delta_j|C_{\max}$ if task processing times are inversely proportional to the number of assigned processors, $p_j(r) = p_j/r$. It is denoted as $P|spdp - any|C_{\max}$ if processing times are arbitrary functions of the number of processors.

Off- and on-line heuristics with performance guarantees are often developed for multiprocessor task scheduling problems, see, for example, Wang and Cheng [61, 60], Choundhary et al. [18], Srinivasa Prasanna and Musicus [52], Blazewicz et al. [11], Blazewicz et al. [9], Dutot et al. [24] and Hunold [30].

The relation with multiprocessor task scheduling allows establishing the computational complexity of the following special cases of the problem $Feasible(Q)$, in which a single operation is assigned to each station: 1) $p_j(r) = 1$ for any r, $a_j = b_j$, $j \in N$; 2) $p_j(r) = 1$ for any r,

$a_j = b_j$, $b_j \in \{1, \ldots, \Delta\}$, $j \in N$, Δ is a given constant; 3) $Q = 5$, $a_j = b_j$, $j \in N$; 4) $Q \in \{2, 3\}$, $a_j = b_j$, $j \in N$; 5) $a_j = 0$, $b_j = Q$, $p_j(r) = p_j/g_j(r)$, $g_j(r)$ is a convex increasing function; 6) $a_j = b_j = 1$, $j \in N$. Case 1) is strongly NP-hard, because, in the notation of Drozdowski [23], problem $P|size_j, p_j = 1|C_{\max}$ is strongly NP-hard due to Lloyd [39]. Case 2) is solvable in $O(n)$ time, because problem $P|size_j \in \{1, \ldots, \Delta\}, p_j = 1|C_{\max}$ is solvable in $O(n)$ time due to Blazewicz et al. [10]. Case 3) is strongly NP-hard and case 4) is pseudo-polynomially solvable, because problem $P5|size_j|C_{\max}$ is strongly NP-hard and problem $Pm|size_j|C_{\max}$, $m \in \{2, 3\}$, is pseudo-polynomially solvable due to Du and Leung [25]. Case 5) is solvable in $O(n)$ time, because it reduces to the malleable task scheduling problem studied by Blazewicz et al. [8] and Barketau et al. [6]. In an optimal solution of the latter problem, each task is assigned to all available processors. Case 6) is strongly NP-hard, because it is equivalent to the decision version of the classic scheduling problem $P||C_{\max}$, which is NP-hard in the strong sense due to Garey and Johnson [27].

We now prove that the problem *MinNumber* is NP-hard in the strong sense if each station is composed of only one operation, values a_j and b_j differ by one unit, and operation processing times are inversely proportional to the number of assigned workers, as they are in the industrial case of the project amePLM [1], which motivates our studies.

Theorem 1 *Problem MinNumber is NP-hard in the strong sense if a single operation is assigned to each station, $a_j = b_j - 1$ and $p_j(r) = p_j/r$, $j \in N$.*

Proof: We use a reduction from the NP-complete problem 3-Partition, see Garey and Johnson [27].

3-Partition: Given $3v+1$ positive integer numbers $h_1, \ldots, h_{3v}$ and H satisfying $\sum_{j=1}^{3v} h_j = vH$, does there exist a partition of the set $\{1, \ldots, 3v\}$ into subsets $Y_1, \ldots, Y_v$ such that $\sum_{j \in Y_t} h_j = H$ for $t = 1, \ldots, v$? Assume without loss of generality that $h_j \geq v+1$, $j = 1, \ldots, 3v$. Otherwise, all numbers $h_1, \ldots, h_{3v}$ and H can be multiplied by $v + 1$ without any influence on the problem's complexity.

Using an instance of 3-Partition, construct an instance of the problem *MinNumber*, in which there are $n = 3v$ operations and the same number of stations, operation j is assigned to station j, $p_j(r) = h_j/r$, $a_j = h_j - 1$, $b_j = h_j$, $j = 1, \ldots, n$, and $d = v$. Thus, the processing time of operation j can take one of the two values: 1 or $1 + \frac{1}{h_j - 1}$, where $1 \leq 1 + \frac{1}{h_j - 1} \leq 1 + \frac{1}{v}$, $j = 1, \ldots, n$. Note that any pair of operations can be performed in parallel if they are assigned to different workers. We will prove that a feasible solution for this instance with value $W^*_{\max} \le H$ exists if and only if the original instance of 3-Partition has a solution.

Suppose that the problem *MinNumber* has a feasible solution with r_j workers assigned to operation j, $j = 1, \dots, n$, and the maximal number of workers employed simultaneously in the line $W^*_{\max} \le H$. In order to illustrate this solution, let us represent an assignment of a worker i to operation j as a *small rectangle* of length $p_j(r_j)$ and height 1, located in line i of a *large rectangle* of length v and height H. Thus, the solution of $MinNumber$ can be viewed as a large rectangle with a set of small rectangles inscribed into it, so that small rectangles do not overlap one another. Figure 2 can serve as an example of such representation. Since $rp_j(r) = h_j$ for any r, each operation j adds h_j to the total area of the large rectangle regardless of the workforce assignment. The union of small non-overlapping rectangles must form the large rectangle, because $\sum_{j=1}^n h_j = vH$.

Now we will prove that every operation j is assigned h_j workers. Assume the contrary: there exists at least one operation q with $h_q - 1$ workers assigned to it. Consider one of these workers. Let he or she be employed to perform w operations. Let J be a subset of w. Each operation $j \in J$ is assigned $h_j - 1$ workers. Since $w = v$, we have $w \le v - 1$. Knowing that any operation must be processed for at least one time unit, and the duration of at least one operation exceeds one time unit, we conclude that this worker's total working time is greater than v. Therefore, the cycle time is exceeded. The total working time of this worker is equal to

$$w + \sum_{j \in J} \frac{1}{h_j - 1} \le v - 1 + \frac{|J|}{v} \le v - 1 + \frac{w}{v} \le v - 1 + \frac{v-1}{v} < v.$$

This strict inequality means that there is a space in the large rectangle, which is not filled in by any small rectangle. This is a contradiction. Therefore, each operation j is assigned h_j workers. As a result, processing times of all operations are equal to 1.

Let Y_t be the set of operations executed simultaneously in the time interval $[t-1, t]$. Since the union of small rectangles forms the large rectangle, equality $\sum_{j \in Y_t} h_j = H$ holds for $t = 1, \dots, v$, as required for the part "only if" of the proof.

Part "if" is trivial. If $Y_1, \dots, Y_v$ is a solution of the problem 3-Partition, then assign h_j workers to operation j, $j = 1, \dots, n$, and execute operations of the set Y_t simultaneously in the time interval $[t-1, t]$, $t = 1, \dots, v$. ■

Computational complexity results are summarized in Table 2.

Table 2: Complexity of special cases of $MinNumber$, in which a single operation is assigned to each station

Problem characteristics	Complexity
$p_j(r) = 1$, $a_j = b_j$	strongly NP-hard
$p_j(r) = 1$, $a_j = b_j$, $b_j \in \{1, \ldots, \Delta\}$, Δ is a given constant	$O(n \log r_{\max})$
$r_{\max} = 5$, $a_j = b_j$	strongly NP-hard
$r_{\max} \in \{2, 3\}$, $a_j = b_j$	pseudo-polynomially solvable
$p_j(r) = p_j/g_j(r)$, $g_j(r)$ is convex increasing, $a_j = 0$, $b_j = r_{\max}$	$O(n \log r_{\max})$
$a_j = b_j = 1$	strongly NP-hard
$p_j(r) = h_j/r$, $a_j = h_j - 1$, $b_j = h_j$	strongly NP-hard

5 Heuristics

Besides the exact bisection search procedure, we also use three parametrized constructive heuristics: two conventional ones and one randomized. All the heuristics use a numerical parameter α, $0 \le \alpha \le 1$, which affects their behavior in such a way that a higher value of α tends to assign workers with larger *ready times* to a selected operation.

Conventional heuristics $TopLong(\alpha)$ and $TopLongPath(\alpha)$

Step 1 Determine an initial hypothetical minimal total number of workers, W, such that $\max_{j\in N}\{a_j\} \le W \le r_{\max}$. A specialist in production planning can provide the initial value of W. Otherwise, we can set $W = \max\left\{ \max_{j\in N}\{a_j\}, \left\lceil \sum_{j\in N} p_j(b_j)/d \right\rceil \right\}$.

Set $r_j = a_j$ and calculate $p_j(r_j)$ for $j = 1, \ldots, n$. Initiate *ready times* T_i *of workers*: $T_i = 0$, $i = 1, \ldots, W$, and *ready times* t_j *of operations*: $t_j := 0$, $j = 1, \ldots, n$.

Step 2 Identify the set N^+ of operations without predecessor in graph G. In heuristic $TopLongPath(\alpha)$, calculate the *longest path* P^* connecting a vertex of N^+ with any other vertex of graph G. Path length is the total *weight* of its vertices, and the weight of vertex j is $p_j(r_j)$.

Step 3 In heuristic $TopLong(\alpha)$ select $j^* \in N^+$ with the largest value $p_j(r_j)$. In heuristic $TopLongPath(\alpha)$ select j^* as the first vertex of the longest path P^*.

Let the workers be ordered such that $T_{i_1} \le \cdots \le T_{i_W}$. Determine sets of r_{j^*} workers $X_h := \{i_{h-r_{j^*}+1}, i_{h-r_{j^*}+2}, \ldots, i_h\}$, $h = r_{j^*}, r_{j^*}+1, \ldots, k$, where k satisfies $T_{i_k} \le \alpha T_{i_W} + (1-\alpha)T_{i_{r_{j^*}}}$ and $T_{i_{k+1}} > \alpha T_{i_W} + (1-\alpha)T_{i_{r_{j^*}}}$, $T_{i_{W+1}} := +\infty$. For example, $X_k = \{i_1, \ldots, i_{r_{j^*}}\}$ if $\alpha = 0$

and $X_k = \{i_{W-r_{j^*}+1}, \dots, i_W\}$ if $\alpha = 1$. Select index h which minimizes $\max\{t_{j^*}, T_{i_h}\} - T_{h-r_{j^*}+1}$ for $r_{j^*} \le h \le k$. Let it be index h^*.

Assign workers of the set X_{h^*} to operation j^* so that all of them start performing this operation at the same time $t^* := \max\{t_{j^*}, T_{i_{h^*}}\}$. Note that index h^* is selected such that the maximum *idle time* of workers in the sets X_h, $h = r_{j^*}, r_{j^*}+1, \dots, k$, just before they start processing operation j^*, is minimized.

Update ready times of workers so that $T_h := t^* + p_{j^*}(r_{j^*})$, $h \in X_{h^*}$. Update ready times of *immediate successors* of vertex j^* so that $t_j := \max\{t_j, t^*\} + p_{j^*}(r_{j^*})$, $j \in A(j^*)$, where $A(j^*)$ is the set of immediate successors of j^* in graph G. Update graph G by removing vertex j^* from it.

If G is empty, then a complete schedule is constructed and the following computations are performed:

1. Determine the makespan $C_{\max} = \max\{T_i \mid i = 1, \dots, W\}$.
2. If the line cycle time is not violated, $C_{\max} \le d$, then return the constructed feasible schedule with W workers and stop.
3. Suppose that $C_{\max} > d$.
   - If $W = r_{\max}$, then return the infeasible schedule with $r_{\max}$ workers and stop. Even with the maximal possible number of workers, the heuristic could not find a feasible schedule.
   - If $W \le r_{\max} - 1$, a feasible schedule still may exist. Determine operations of a *critical path* that do not intersect in time and whose processing times sum up to $C_{\max}$. Reset $r_j := \min\{r_j + 1, b_j, W + 1\}$ for every such operation, reset $W := W + 1$, restore original graph G, and go to Step 2.

If G is not empty, then perform Step 2. ■

There are three possible outputs of heuristics $TopLong(\alpha)$ or $TopLongPath(\alpha)$:

- Feasible schedule is obtained.
- Infeasible schedule with $r_{\max}$ workers is found.
- Solving time limit is reached. The last found infeasible schedule is returned.

If the solution time limit permits, then heuristics $TopLong(\alpha)$ and $TopLongPath(\alpha)$ can be run for several values of α, $0 \leq \alpha \leq 1$.

The **randomized heuristic** $TopRandom(\alpha)$ is similar to $TopLong(\alpha)$ with only one difference. At Step 3 of $TopRandom(\alpha)$ operation $j^* \in N^+$ is selected at random. Heuristic $TopRandom(\alpha)$ can be run several times and for several values of α until the solving time limit is reached. Computational experiments with the heuristics are described in Section 6.

6 Computational study

We performed two series of computational experiments with the problem $MinNumber$. The goal of the first series is to establish the maximum problem size in terms of the number of operations that can be solved to optimality in a reasonable time based on the bisection search and the MILP formulation. The goal of the second series is to compare the quality of the heuristics. We used the data generator for the simple assembly line balancing problem described in Otto et al. [47], extending it to match the specificity of the problem $MinNumber$. This generator constructs instances with the number of operations $n = 20$ and $n = 50$. We do the same. The number of stations, the assignment of operations to the stations and the box constraints are selected to conform with the real life problem of our industrial partner, see Battaïa et al. [7]. Processing times are set to $p_j(1) = p_j$, where p_j are generated as in [47], and $p_j(r) = p_j/r$ for $r \geq 2$. We assumed that each connected component of a precedence graph, generated as in [47], corresponds to a station. Values $a_j \in \{1, 2\}$ are generated with probability $2/3$ for $a_j = 1$ and probability $1/3$ for $a_j = 2$, and values $b_j = 4$ for all j, as in the industrial application. The source code of the experiment is available on request at GitHub [2].

6.1 Exact solution: maximum number of operations

The MILP model $Feasible(Q)$ was handled by the solver IBM ILOG CPLEX Optimization Studio 12.6.2, which was run on a computer Intel Xeon with CPU E5-2673 v3 2.4 GHz, 8 GB of RAM, MS Windows Server 2008 R2 Datacenter 64bit and 4 or 16 treads. The time limit for solving each instance was set to one hour.

For $n = 50$, no instance of $MinNumber$ was solved to optimality within one hour in either configuration. For $n = 20$, 250 instances were generated. Since functions $p_j(r) = p_j/r$ are convex, the relaxed problem with $a_j = 1$, $b_j = W^*_{\max}$, no precedence constraints and the possibility to perform any operations in parallel if they are assigned to different workers, is the special case 5) in Section 4, which is solved by assigning all $W^*_{\max}$ workers to each operation. Therefore, $W^*_{\max} \geq LB = \left\lceil \sum_{i \in N} p_i/d \right\rceil$ in this case, and we used the latter lower bound LB in the experiments. We calculated the upper bound UB on the number of workers as the sum of the minimal numbers of workers required for each station independently: $UB = \sum_{k=1}^{m} \max \left\{ \max_{j \in N_k} \{a_j\}, \left\lceil \sum_{j \in N_k} p_j/d \right\rceil \right\}$.

Recall that the bisection search procedure includes solving the problem $Feasible(Q)$ for $Q = LB$, where LB depends on the problem instance as explained in the previous paragraph. Fig. 3 shows the numbers of solved and unsolved instances of the problem $Feasible(LB)$ for $n = 20$. The horizontal axis represents the values of LB that were calculated for all the generated instances. The height of the color column corresponding to a given LB represents the number of instances with this LB. One can see that the number of solved instances decreases as the number of workers increases.

We observed that the solver finds a feasible solution of the problem $Feasible(Q)$ quite fast if it exists, but it takes a lot of time to detect infeasibility. This observation can be used to solve the problem $Feasible(Q)$ heuristically: if a feasible solution is not found within a certain time limit, then it is decided that the problem $Feasible(Q)$ has no solution. This observation is confirmed by the fact that the ratio of the number of solved instances to the number of unsolved instances increases as the value of Q increases from LB to UB.

We also noticed that the idle time parameter calculated as $Idle = \frac{Qd - \sum_{j \in N} p_j}{Qd}$ impacts the performance, see Fig. 4 for an example. One can see that the higher values of $Idle$ imply the smaller solution time of the problem $Feasible(3)$.

We compared the impact of the number of available processor threads, 4 or 16, on the first 100 generated instances. Table 3 shows that using 16 processor threads instead of 4 leads to solving 7% more instances within the one hour time limit. However, the average

Configuration	4 threads	16 threads
Percent of solved instances	67%	75% (+7%)
Average solution time of solved instances	225 sec	271 sec (+46 sec)

Table 3: Scalability of the MILP model

instance solution time for instances solved to optimality slightly increases for the more powerful configuration due to the fact that the instances unsolved for 4 threads also need long solution time for 16 threads. A general observation is that the realization of the MILP model on a larger number of parallel threads does not improve the performance a lot.

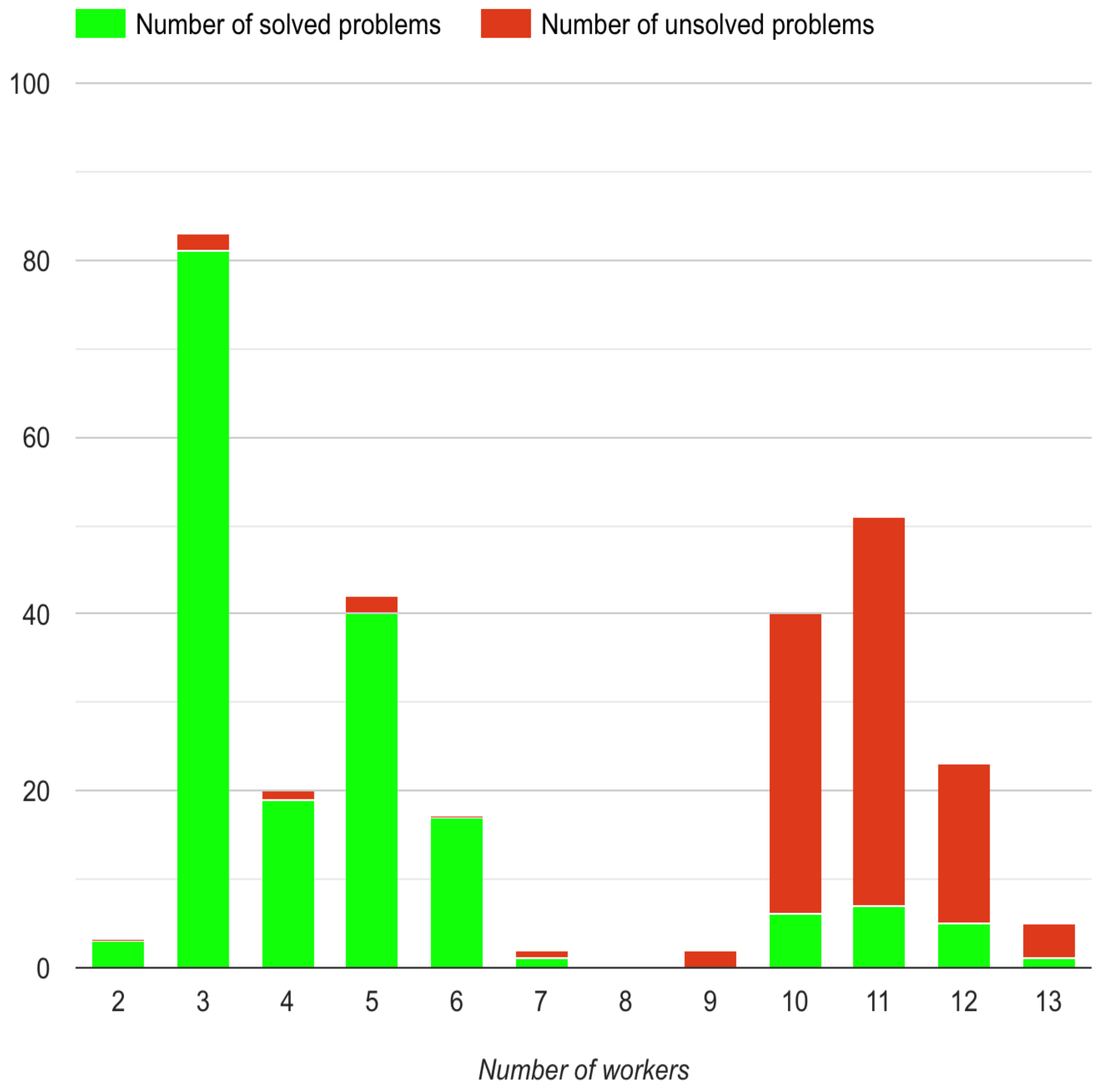


Figure 3: Numbers of solved and unsolved instances of $Feasible(LB)$ in 1 hour for $n = 20$

We made experiments for solving the real life problem from the project amePLM [1] with 170 operations. In this problem, there are 20 production cycles, which differ by the assignment of the same set of operations to the stations. The problem is to minimize the maximum number of workers needed for each cycle. For this problem, the straightforward solution of the MILP model was not possible because of the memory limitation, which is mainly caused by the constraints (6) and (7). We noted that the matrix of these constraints contains many zero entries and decided to modify the model such that it is populated by non-zero entries only. This modification removed the memory problem, but the required CPU time increased dramatically

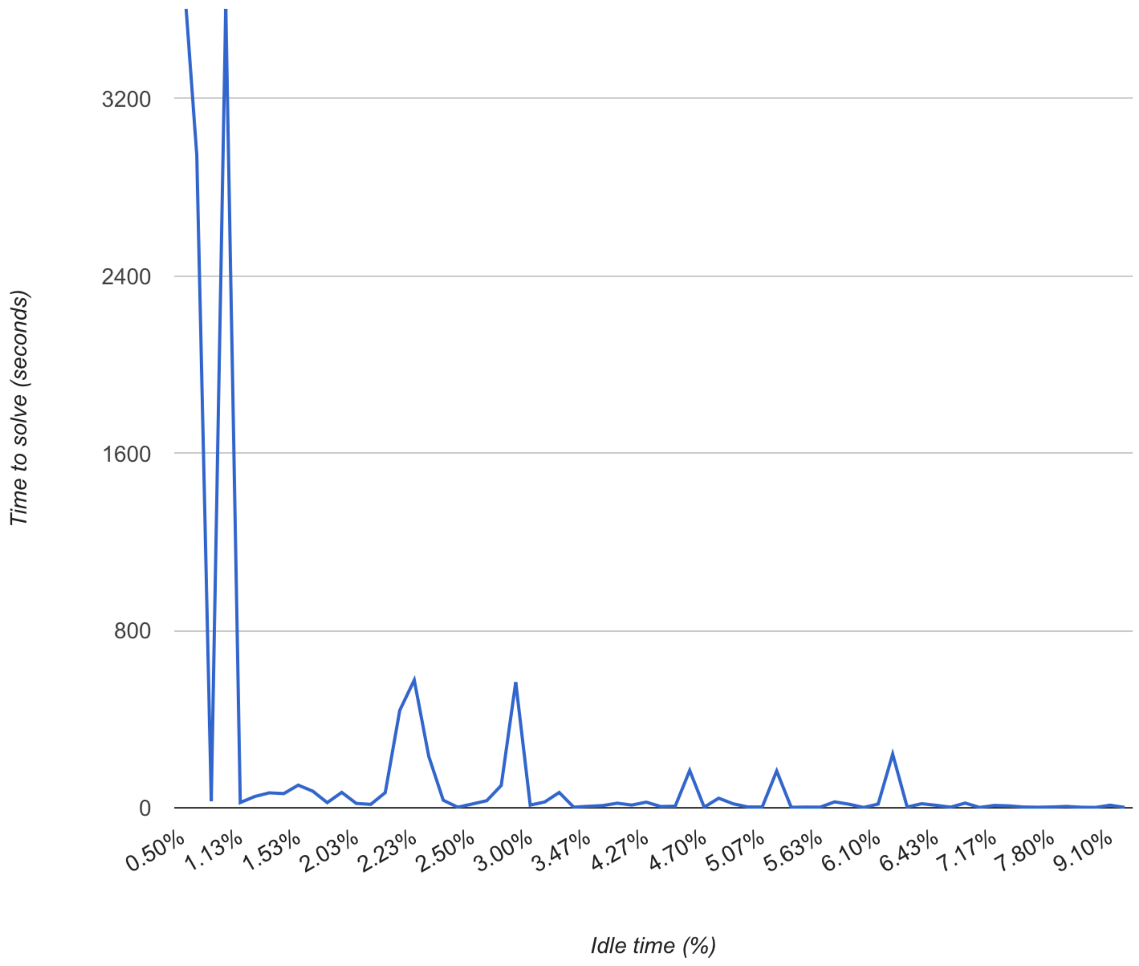
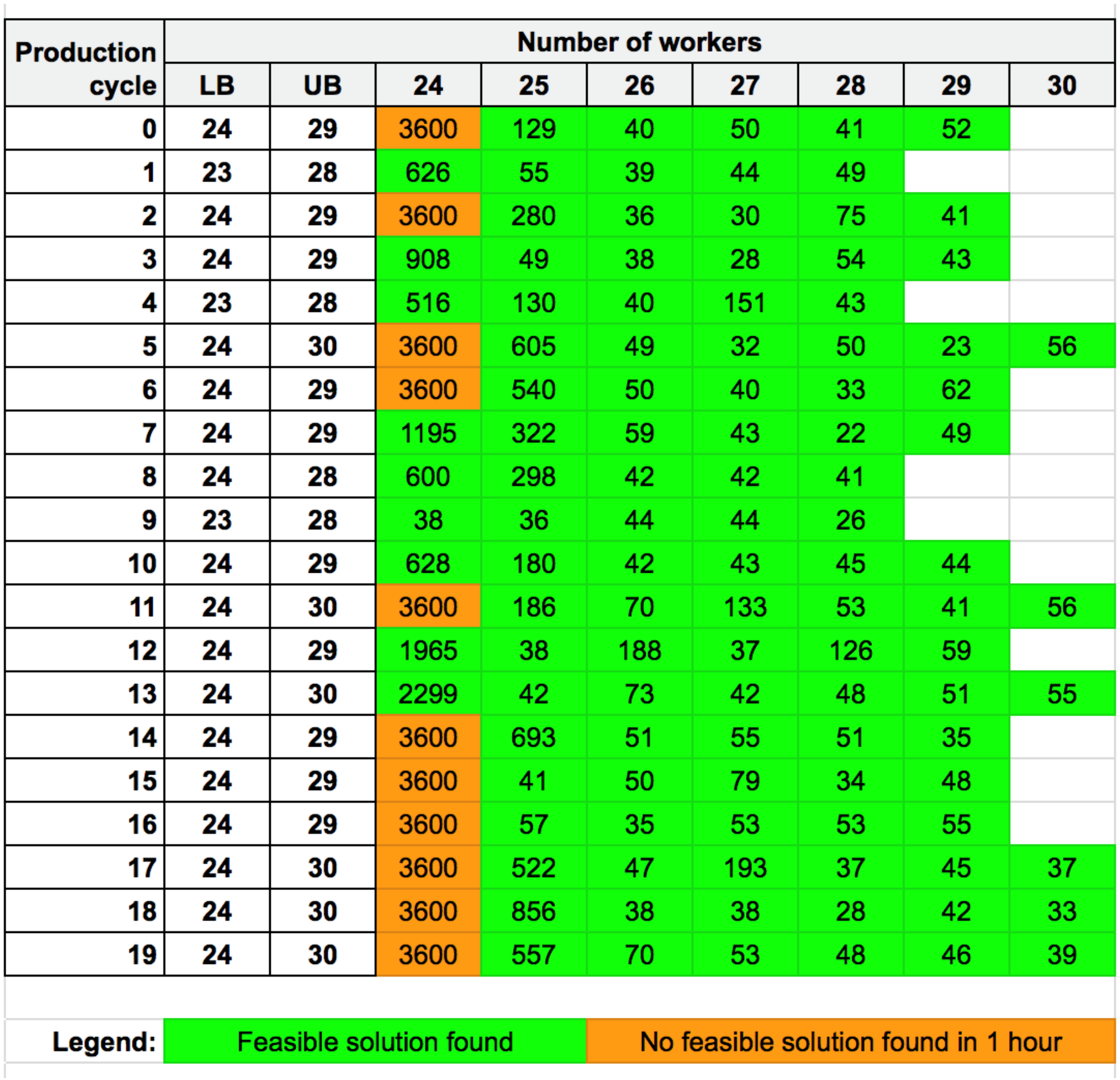


Figure 4: Impact of parameter $Idle$ on solution time of the problem $Feasible(3)$

such that 24 hours were not enough to populate the model. In order to find a compromise between the time and memory requirements, we have tried to balance the model population with non-zero row entries only. However, we failed in all our attempts.

Based on the advice of a referee, we have applied a heuristic approach which is to aggregate operations on the same station, whose intervals $[a_j, b_j]$ intersect, into one operation, say J, associated with the interval $[a_J, b_J]$ being the intersection of the original intervals, and solve the problem with the aggregated operations. By doing this, we reduced the number of operations from 170 to 28. A feasible solution to the problem $MinNumber$ with 25 workers was found in one day by solving the problem $MinNumber$ for each of the 20 production cycles. In order to better demonstrate the performance, we performed experiments with the problems $Feasible(Q)$ for all values of Q from the interval $[LB, UB]$, see Fig. 5. Note that, though $LB = 23$ for several production cycles, the entire line can not operate with less than 24 workers. Therefore, we did



Production	Number of workers								
cycle	LB	UB	24	25	26	27	28	29	30
0	24	29	3600	129	40	50	41	52	
1	23	28	626	55	39	44	49		
2	24	29	3600	280	36	30	75	41	
3	24	29	908	49	38	28	54	43	
4	23	28	516	130	40	151	43		
5	24	30	3600	605	49	32	50	23	56
6	24	29	3600	540	50	40	33	62	
7	24	29	1195	322	59	43	22	49	
8	24	28	600	298	42	42	41		
9	23	28	38	36	44	44	26		
10	24	29	628	180	42	43	45	44	
11	24	30	3600	186	70	133	53	41	56
12	24	29	1965	38	188	37	126	59	
13	24	30	2299	42	73	42	48	51	55
14	24	29	3600	693	51	55	51	35	
15	24	29	3600	41	50	79	34	48	
16	24	29	3600	57	35	53	53	55	
17	24	30	3600	522	47	193	37	45	37
18	24	30	3600	856	38	38	28	42	33
19	24	30	3600	557	70	53	48	46	39

Legend: Feasible solution found | No feasible solution found in 1 hour

Figure 5: Solution time in seconds for the real life instance of the problem $Feasible(Q)$ with $n = 28$ aggregated operations

not solve $Feasible(Q)$ for $Q = 23$. We also did not solve $Feasible(Q)$ for $Q > UB$. The corresponding entries are blank.

A solution with 25 workers for the problem with the 28 aggregated operations was converted into a feasible solution for the original real life problem with 170 operations by addressing the precedence constraints. This result is better than the previously obtained result of 26 workers obtained by Battaïa et al. [7] for the same problem.

6.2 Quality of heuristics

We compared the solution quality of the heuristics $TopLong(\alpha)$, $TopLongPath(\alpha)$ and $TopRandom(\alpha)$. They were tested on the real life instance with 170 operations. We made experiments for $\alpha \in \{0, 0.1, 0.2, \ldots, 1\}$. The heuristic TopRandom(α) was applied in two scenarios, with 100 and 1000 runs for each α. The obtained results are demonstrated in Fig. 6. 1000 runs of TopRandom(α) for a given α take about 30 minutes and provide better result

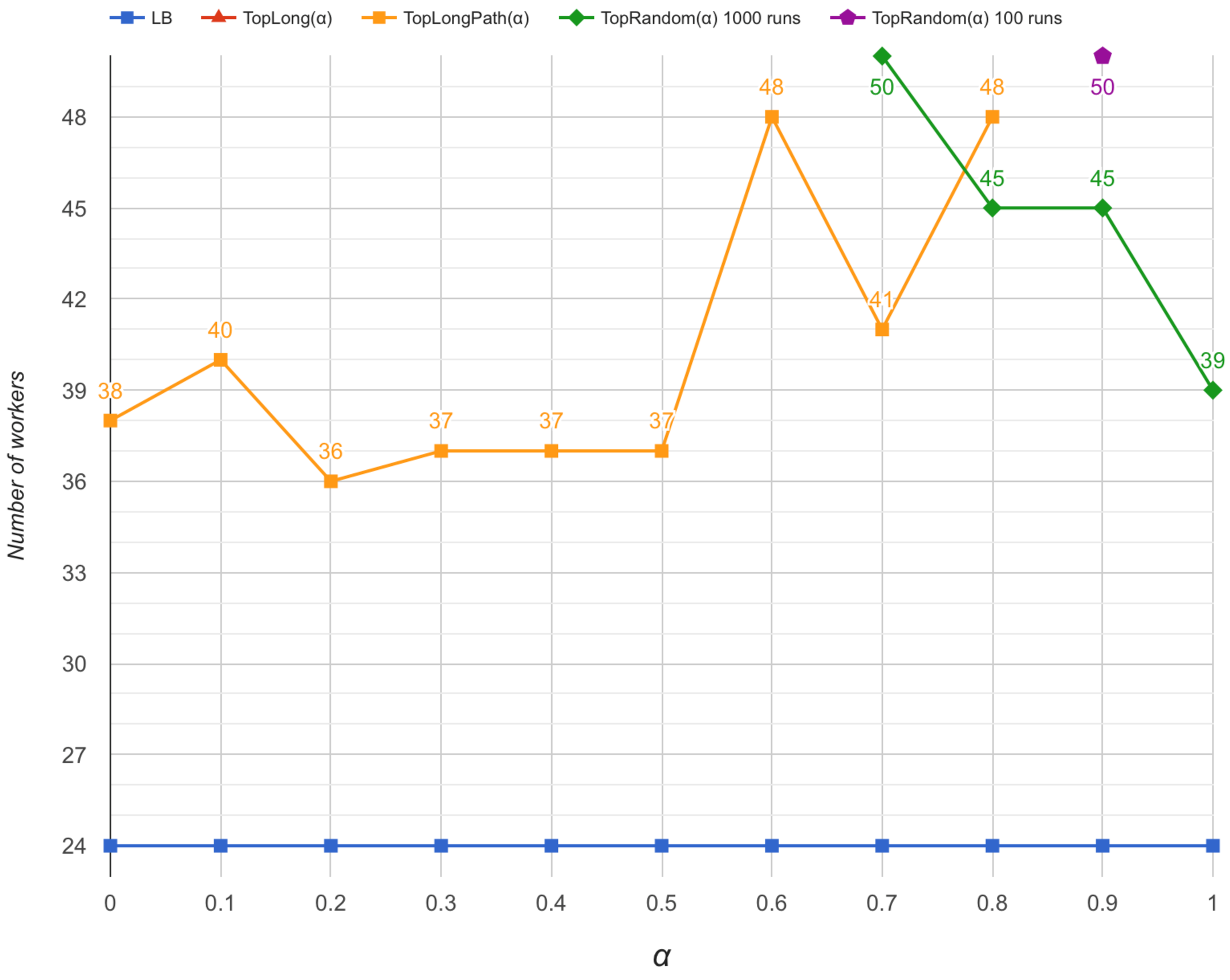


Figure 6: Heuristic solutions for the real life instance with $n = 170$

than 100 runs of the same heuristic. Heuristic $TopLong(\alpha)$ is not mentioned in Fig. 6 because it finds unreasonably high numbers of workers for all values of α.

We also applied heuristics to solve the problem with the 28 aggregated operations. TopRandom(α) was run 100000 times for each value of α, and all these runs required less than 30 minutes. The results are given in Figure 7.

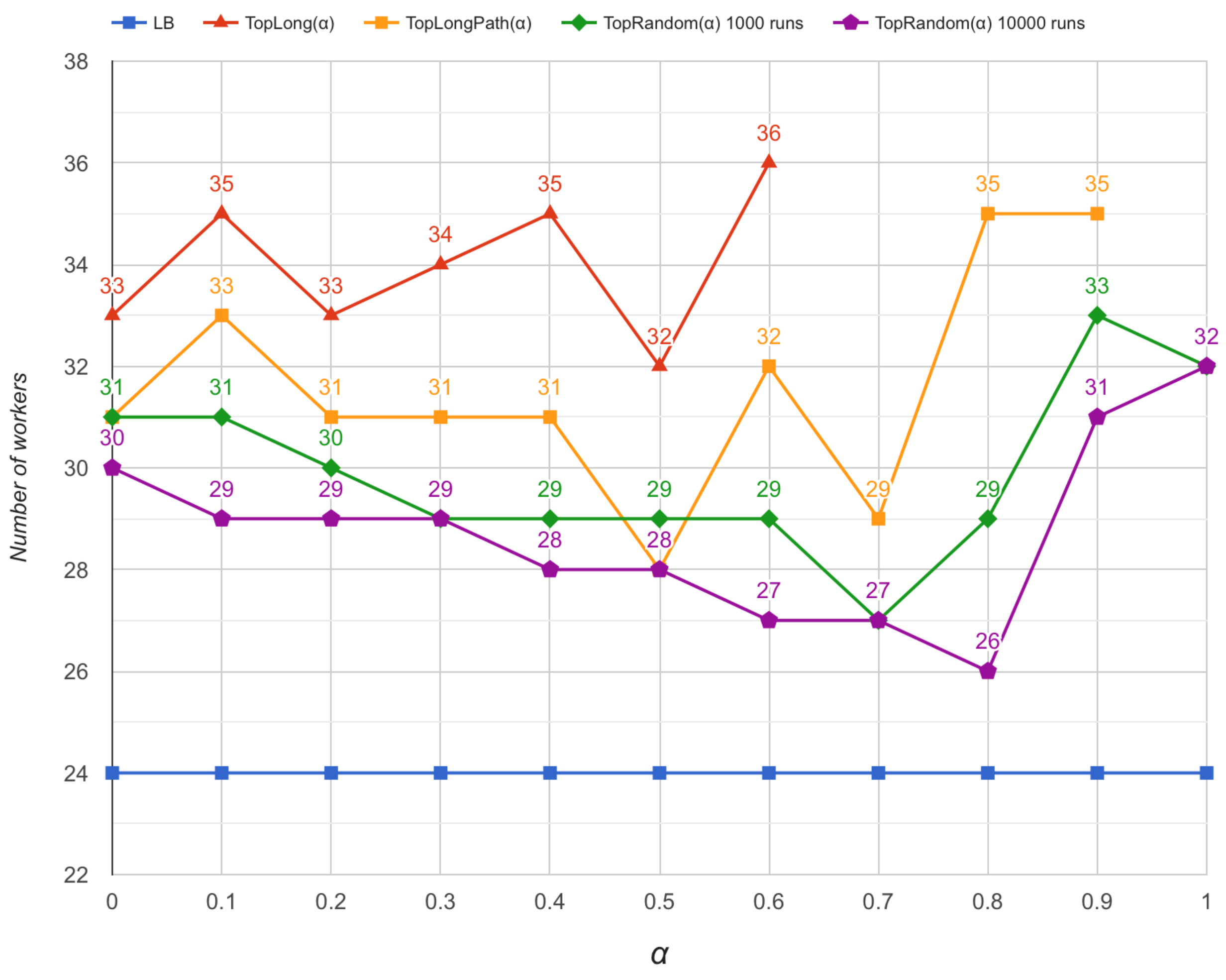


Figure 7: Heuristic solutions for the real life instance with $n = 28$ aggregated operations

We noticed that the heuristics were able to find a solution 10000 times faster than the MILP solver for the same problem $Feasible(Q)$. The minimal number of workers delivered by all the heuristics can be used as an upper UB in the exact bisection search procedure for the problem $MinNumber$.

7 Conclusions and suggestions for future research

The following results are obtained for the problem $MinNumber$: two conventional and one randomized heuristics, a bisection search reduction to a series of feasibility problems $Feasible(Q)$, a MILP model for the feasibility problem, a relation of the feasibility problem to the multi-mode project scheduling problems and multiprocessor moldable task scheduling problems, which is used to establish computational complexity of several special cases of the problem $MinNumber$, and computer experiments with the suggested exact and heuristic solution approaches. The number of workers in the real life problem of the project amePLM is decreased from the earlier obtained number of 26 to 25.

Computational experiments demonstrated that the instances with up to 20 operations and 10 workers can be solved to optimality in a reasonable time on a standard computer. They also showed that the quality of the heuristic solutions for the industrial problem with 170 operations is sufficiently good. The results can also be useful for modeling and solving relevant multi-mode project scheduling problems and moldable task scheduling problems.

Developing metaheuristic and matheuristic approaches, which are able to find near optimal solutions of the problem $MinNumber$, is interesting from the practical point of view, and investigating computational complexity of special cases of this problem is interesting from the theoretical point of view. For example, what is the complexity of the problem $MinNumber$, in which $p_j(r) = p/r$ and $r_j \in \{a, b\}$, $j \in N$, for given p, a and b?

The studied problem can be extended by relaxing the assumption that the number of stations and the assignment of operations to the stations are fixed. The relationship between the number of stations and the minimum number of workers $W^*_{\max}$ in this setting would be of significant interest.

References

[1] `http://cordis.europa.eu/project/rcn/100702_en.html`

[2] `http://malyutins.github.io/Workforce_minimization/`

[3] Akagi, F., Osaki, H., Kikuchi, S., 1983. A method for assembly line balancing with more than one worker in each station. International Journal of Production Research 21(5), 755-770.

[4] Araújo, F.F.B., Costa, A.M., Miralles, C., 2012. Two extensions for the ALWABP: Parallel stations and collaborative approach. International Journal of Production Economics 140(1), 483-495.

[5] Artigues, C., Demassey, S., Néron, E., 2013. Resource-constrained project scheduling: models, algorithms, extensions and applications. ISTE Ltd and John Wiley & Sons.

[6] Barketau, M., Kovalyov, M.Y., Weglarz, J., Machowiak, M., 2014. Scheduling arbitrary number of malleable tasks on multiprocessor systems. Bulletin of the Polish Academy of Sciences Technical Sciences 62(2), 255-261.

[7] Battaïa, O., Delorme, X., Dolgui, A., Malyutin, S., Horlemann, A., Kovalev, S., 2015. Workforce minimization for a mixed-model assembly line in the automotive industry. International Journal of Production Economics 170(B), 489-500.

[8] Blazewicz, J., Machowiak, M., Weglarz, J., Kovalyov, M.Y., Trystram, D., 2004. Scheduling malleable tasks on parallel processors to minimize the makespan. Annals of Operations Research 129, 65-80.

[9] Blazewicz, J., Cheng, T.C.E., Machowiak, M., Oguz, C., 2011. Berth and quay crane allocation: a moldable task scheduling model. Journal of the Operational Research Society 62(7), 1189-1197.

[10] Blazewicz, J., Drabowski, M., Weglarz, J., 1986. Scheduling multiprocessor tasks to minimize schedule length. IEEE Transactions on Computers 35(5), 389-393.

[11] Blazewicz, J., Ecker, K., Plateau, B., Trystram, D., 2000. Handbook on parallel and distributed processing, Springer, Berlin.

[12] Besikci, U., Bilge, U., Ulusoy, G., 2013. Resource dedication problem in a multi-project environment, Flexible Services and Manufacturing Journal 25 (1-2), 206-229.

[13] Blum, C., Miralles, C., 2011. On solving the assembly line worker assignment and balancing problem via beam search. Computers & Operations Research 38(1), 328-329.

[14] Borba, L., Ritt, M., 2014. A heuristic and a branch-and-bound algorithm for the Assembly Line Worker Assignment and Balancing Problem. Computers & Operations Research 45, 87-96.

[15] Boysen, N., Emde, S., 2014. Scheduling the part supply of mixed-model assembly lines. European Journal of Operational Research 239(3), 820-829.

[16] Burkard, R.E., Dell'Amico, M., Martello, S., 2009. Assignment problems. SIAM e-books, Philadelphia.

[17] Camm, J.D., Magazine, M.J., Polak, G.G., Zaric, G.S., 2008. Scheduling parallel assembly workstations to minimize a shared pool of labor. IIE Transactions 40(8), 749-758.

[18] Choundhary, A.N., Narahari, B., Nicol, D.M., Simha, R., 1994. Optimal processor assignment for a class of pipelined computations. IEEE Transactions on Parallel and Distributed Systems 5/4, 439-445.

[19] Corominas, A., Pastor, R., Plans, J., 2008. Balancing assembly line with skilled and unskilled workers. Omega 36(6), 1126-1132.

[20] De Bruecker, P., Van den Bergh, J., Beliën, J., Demeulemeester, E., 2015. Workforce planning incorporating skills: State of the art. European Journal of Operational Research 243(1), 1-16.

[21] Demeulemeester, E.L., Herroelen, W.S., Elmaghraby, S.E., 1996. Optimal procedures for the discrete time/cost tradeoff problem in project networks. European Journal of Operational Research, 88(1), 50-68.

[22] Dolgui, A., Kovalev, S., Kovalyov, M.Y., Malyutin, S., Soukhal, A., 2015. Minimizing the number of workers for one cycle of a paced production line. In Preprints of the 15th IFAC Symposium on Information Control Problems in Manufacturing, 2349-2354.

[23] Drozdowski, M., 1996. Scheduling multiprocessor tasks - An overview. European Journal of Operational Research 94(2), 215-230.

[24] Dutot, P.-F., Mounié, G., Trystram, D., 2004. Scheduling parallel tasks approximation algorithms. In Handbook of Scheduling: Algorithms, Models, and Performance Analysis, edited by Joseph Y.-T. Leung, CRC Press, LLC.

[25] Du, J., Leung, J.Y-T., 1989. Complexity of scheduling parallel task systems. SlAM Journal on Discrete Mathematics 2(4), 473-487.

[26] Ernst, A.T., Jiang, H., Krishnamoorthy, M., Sier, D., 2004. Staff scheduling and rostering: A review of applications, methods and models. European Journal of Operational Research, 153 (1), 3-27.

[27] Garey, M.R., Johnson, D.S., 1979. Computers and intractability: A guide to the theory of NP-completeness. Freeman, San Francisco.

[28] Ghoddousi, P., Eshtehardian, E., Jooybanpour, S., Javanmardi, A., 2013. Multi-mode resource-constrained discrete time-cost-resource optimization in project scheduling using non-dominated sorting genetic algorithm. Automation in Construction 30, 216-227.

[29] Giard, V., Jeunet, J., 2010. Optimal sequencing of mixed models with sequence-dependent setups and utility workers on an assembly line. International Journal of Production Economics 123(2), 290-300.

[30] Hunold, S., 2015. One step toward bridging the gap between theory and practice in moldable task scheduling with precedence constraints. Concurrency Computation 27(4), 1010-1026.

[31] Karabak, F., Güner, N.D., Satir, B., Kandiller, L., Gürsoy, I., 2011. An optimization model for worker assignment of a mixed model vehicle production assembly line under worker mobility. In Proceedings of the 41st International Conference on Computers & Industrial Engineering, 483-490.

[32] Kolisch, R., Sprecher, A., 1997. PSPLIB - A project scheduling problem library: OR Software - ORSEP Operations Research Software Exchange Program. European Journal of Operational Research 96(1), 205-216.

[33] Kolisch, R., Sprecher, A., Drexl, A., 1995. Characterization and generation of a general class of resource-constrained project scheduling problems. Management Science, 41(10), 1693-1703.

[34] Koltai, T., Tatay, V., 2013. Formulation of workforce skill constraints in assembly line balancing models. Optimization and Engineering 14(4), 529-545.

[35] Kovalyov, M.Y., Delorme, X., Dolgui, A., 2016. Workforce planning for cyclic production of multiple parts. The First International Workshop on Dynamic Scheduling Problems, Poznan.

[36] Kouvelis, P., Karabati, S., 1999. Cyclic scheduling in synchronous production lines, IIE Transactions 31(8), 709-719.

[37] Lee, C.-Y., Vairaktarakis, G.L., 1997. Workforce planning in mixed model assembly systems. Operations Research 45(4), 553-567.

[38] Li, H., Womer, K., 2012. Optimizing the supply chain configuration for make-to-order manufacturing. European Journal of Operational Research 221(1), 118-128.

[39] Lloyd, E.L., 1981. Concurrent task systems. Operations Research 29(1), 189-201.

[40] Lutz, C.M., Davis, K.R., 1994. Development of operator assignment schedules: a DSS approach. Omega 22(1), 57-67.

[41] Miller, J.C., 2013. Fundamentals of shiftwork scheduling, 3rd Edition: Fixing Stupid. Smashwords.

[42] Monma, C.L., Schrijver, A., Todd, M.J., Wei, V.K., 1990. Convex resource allocation problems on directed acyclic graphs: duality, complexity, special cases, and extensions. Mathematics of Operations Research 15(4), 736-748.

[43] Moreira, M.C.O., Costa, A.M., 2013. Hybrid heuristics for planning job rotation schedules in assembly lines with heterogeneous workers. International Journal of Production Economics 141(2), 552-560.

[44] Moreira, M.C.O., Cordeau, J.-F., Costa, A.M., Laporte, G., 2015. Robust assembly line balancing with heterogeneous workers. Computers & Industrial Engineering 88, 254-263.

[45] Mutlu, O., Polat, O., Supciller, A.A., 2013. An iterative genetic algorithm for the assembly line worker assignment and balancing problem of type II. Computers & Operations Research 40(1), 418-426.

[46] Naveh, Y., Richter, Y., Altshuler, Y., Gresh, D.L., Connors, D.P., 2007. Workforce optimization: Identification and assignment of professional workers using constraint programming. IBM Journal of Research and Development 51(3/4), 263-279.

[47] Otto, A., Otto, C., Scholl, A., 2013. Systematic data generation and test design for solution algorithms on the example of SALBPGen for assembly line balancing. European Journal of Operational Research 228(1), 33-45.

[48] Ranjbar, M.R., Kianfar, F., 2007. Solving the discrete time/resource trade-off problem in project scheduling with genetic algorithms. Applied Mathematics and Computation 191(2), 451-456.

[49] Ritt, M., Costa, A.M., Miralles, C., 2016. The assembly line worker assignment and balancing problem with stochastic worker availability. International Journal of Production Research 54(3), 907-922.

[50] Rocha, M., Oliveira, J.F., Carravilla, M.A., 2012. Quantitative approaches on staff scheduling and rostering in hospitality management: An overview. American Journal of Operations Research, 2, 137–145.

[51] Salewski, F., Schirmer, A., Drexl, A., 1997. Project scheduling under resource and mode identity constraints: Model, complexity, methods, and application. European Journal of Operational Research 102(1), 88-110.

[52] Srinivasa Prasanna, G.N., Musicus, B.R., 1994. Generalized multiprocessor scheduling for directed acyclic graphs. In: Proceedings of Supercomputing 1994, IEEE Press, New York, 237-246.

[53] Sungur, B., Yavuz, Y., 2015. Assembly line balancing with hierarchical worker assignment. Journal of Manufacturing Systems 37(1), 290-298.

[54] Tseng, L.Y., Chen, S.C., 2009. Two-phase genetic local search method for multimode resource-constrained project scheduling problem. IEEE Transactions on Evolutionary Computation 13(4), 848-857.

[55] Vairaktarakis, G.L., Cai, X., 2003. Complexity of workforce scheduling in transfer lines. Journal Journal of Global Optimization 27(2-3), 273-291.

[56] Vairaktarakis, G.L., Cai, X., Lee, C.-Y., 2002. Workforce planning in synchronous production systems. European Journal of Operational Research 136(1), 551-572.

[57] Vairaktarakis, G.L., Winch, J.K., 1999. Worker cross-training in paced assembly lines, Manufacturing and Service Operations Management 1(2), 112-131.

[58] Vidic, N.S., 2008. Developing methods to solve the workforce assignment problem considering worker heterogeneity and learning and forgetting. Doctoral Dissertation, University of Pittsburgh.

[59] Vilà, M., Pereira, J., 2014. A branch-and-bound algorithm for assembly line worker assignment and balancing problems. Computers & Operations Research 44, 105-114.

[60] Wang, Q., Cheng, K.H., 1992. A heuristic of scheduling parallel tasks and its analysis. SIAM Journal on Computing 21(2), 281-294.

[61] Wang, Q., Cheng, K.H., 1991. List scheduling of parallel tasks. Information Processing Letters 37(5), 291-297.

[62] Willemen, R.J., 2002. School timetable construction: algorithms and complexity. Technische Universiteit Eindhoven.

[63] Wilson, J.M., 1986. Formulation of a problem involving assembly lines with multiple manning of work stations, International Journal of Production Research 24(1), 59-63.