Workforce minimization for a mixed-model assembly line in the automotive industry

Olga Battaïa, Xavier Delorme, Alexandre Dolgui, Johannes Hagemann, Anika Horlemann, Sergey Kovalev, Sergey Malyutin

Workforce minimization for a mixed-model assembly line in the automotive industry

Olga Battaïa [a], Xavier Delorme [a], Alexandre Dolgui [a,*], Johannes Hagemann [b], Anika Horlemann [b], Sergey Kovalev [c,1], Sergey Malyutin [a]

[a] LIMOS, UMR CNRS 6158, Ecole des Mines de Saint-Etienne, 158 cours Fauriel, 42023 Saint-Etienne Cedex 2, France
[b] MBtech Group GmbH & Co. KGaA, Germany
[c] INSEEC Business School ECE Lyon, 19 Place Tolozan, 69001 Lyon, France

Keywords:
Assembly line
Workforce dimensioning
Worker assignment
Scheduling
Optimization
Integer linear programming
Heuristics

A B S T R A C T

A paced assembly line consisting of several workstations is considered. This line is intended to assemble products of different types. The sequence of products is given. The sequence of technological tasks is common for all types of products. The assignment of tasks to the stations and task sequence on each station are known and cannot be modified, and they do not depend on the product type. Tasks assigned to the same station are performed sequentially. The processing time of a task depends on the number of workers performing this task. Workers are identical and versatile. If a worker is assigned to a task, he/she works on this task from its start till completion. Workers can switch between the stations at the end of each task and the time needed by any worker to move from one station to another one can be neglected. At the line design stage, it is necessary to know how many workers are necessary for the line. To know the response to this question we will consider each possible takt and assign workers to tasks so that the total number of workers is minimized, provided that a given takt time is satisfied. The maximum of minimal numbers of workers for all takts will be considered as the necessary number of workers for the line. Thus, the problem is to assign workers to tasks for a takt. We prove that this problem is NP-hard in the strong sense, we develop an integer linear programming formulation to solve it, and propose conventional and randomized heuristics.

1. Introduction

The current highly competitive market forces manufacturing based industries to optimize their production costs. Workforce minimization in manual assembly lines is one of the challenging issues. This paper proposes models and methods to help the practitioners to dimension and schedule workforce resources on mass production manual mixed-model assembly lines.

On manual mixed-model lines, not only one but a set of similar products (variants or models) are assembled. Usually, for mixed-model assembly line balancing, a set of tasks for each variant is assigned to each workstation on the line and is performed by the worker(s) available at this workstation. Depending on the presence of the different product variants on the line and the processing time required to treat each task of a particular variant at each particular workstation, the distribution of workload among workstations can present considerable inequalities over time. The problem of how to balance the workload among workstations has been intensively studied in the literature (Venkatesh, 2008). However, because of the inequalities of task processing times for different variants, the perfect balance can rarely be attained if at all. This results in idle times and an inefficient use of workforce resources.

Our paper suggests approaching this problem differently. We assume that the assignment of tasks to workstations is known and it is the same for all product variants. It cannot be modified. In contrast, the assignment of workers to workstations can be dynamic depending on the state of the line.

We study the case where a line is designed for a mass production of different variants of a product. The total volume and the ratios of demands for product variants are known. To smooth the production of different variants, a sequence of product variants is defined such that the numbers of product variants in the sequence respect given ratios, and this sequence is repeated cyclically. Considering a given number of product variants and known variant ratios, we can enumerate all possible cyclic sequences of product variants. This number is not so large. In this

* Corresponding author. Tel.: +33 477 42 01 66; fax: +33 477 42 66 66.
E-mail addresses: battaia@emse.fr (O. Battaïa), delorme@emse.fr (X. Delorme), dolgui@emse.fr (A. Dolgui), johannes.hagemann@mbtech-group.com (J. Hagemann), Anika.Horlemann@mbtech-group.com (A. Horlemann), kovalev@emse.fr (S. Kovalev), malyutin@emse.fr (S. Malyutin).
URL: http://www.emse.fr/~dolgui (A. Dolgui).
[1] Tel.: +33 437 92 92 44.

http://dx.doi.org/10.1016/j.ijpe.2015.05.038

paper, we will consider a given sequence of product variants. The approach and model are the same for all other possible sequences.

The assembly line is paced, i.e. at the end of each takt, all items are simultaneously moved to the next stations. The takt time is the time available to execute the tasks assigned to stations.

A state of the line is defined by variants of the product at stations, i.e. if we know which variant is at each station for a takt, we know the state of the line for this takt (and consequently task processing times). For a given cyclic sequence of product variants, all possible states of the line are known and their number (thus the number of takts to consider) is equal to the sequence length. They are repeated cyclically. We will develop a model for only one takt; obviously this model is valid for all other takts, because the takt state structural information is the same for all takts (workers and workstations with their tasks); the differences among takt states concern only processing times which is the input data of our model.

We consider that all workers are polyvalent (identical) and each of them can execute any task at any workstation. The time needed by any worker to move from one station to another one can be neglected. A task is executed by one or several workers. The processing time of a task is inversely proportional to the number of workers performing it. When a task is finished at one workstation, a worker can move to another workstation in order to speed a task completion there and to balance the workloads of both workstations. The assignment of tasks to the stations and the task sequence on each station are known and cannot be modified, and they do not depend on the product variant. Tasks assigned to the same station are performed sequentially. Therefore, for a given takt, the considered optimization problem is to find the optimal assignments of workers to sequences of tasks, so as to find a schedule of their moves along workstations, taking into account the state of the line in the considered takt. The objective is to minimize the total number of workers required for the given takt.

As aforementioned, this problem is the same for any takt; only the states of the line are different, i.e. the product variants are situated differently at stations for diverse takts. Thus, if we have a model to optimize assignment of workers for a takt, this model can be applied to all other takts. The worker moving times are neglected, thus the working positions for different takts can be considered independently. The number of workers assigned to the line cannot be modified from one takt to another one. By applying the model to each possible takt separately, we can obtain a minimum number of workers necessary for this line and a work schedule for each of them. This approach is used to dimension the workforce at the line design stage.

To summarize, we consider one takt scheduling problem for workers for a given sequence of product variants. The objective is to find a schedule of moves of workers along workstations minimizing the number of necessary workers under a takt time constraint and taking into account workloads of stations (the state of the line for the considered takt). This problem is also related to line balancing problems, because it can be considered as an assignment of tasks to workers to minimize the number of workers.

Section 2 presents an industrial case which motivated our research. *We call this problem P.* Production environment and numerical characteristics of the problem are described. Related literature is analyzed in Section 3. A Mixed-Integer Program (MIP) is presented in Section 4. Problem *P* is proved to be NP-hard in the strong sense in Section 5. Five heuristics developed for problem *P* are described in Section 6. Section 7 reports computer implementation and experiments. Finally, the paper concludes with a summary of the results.

2. Industrial case

Problem *P* was stated by one of our industrial partners. This enterprise from the automotive industry has to design a new assembly line for three variants of an engine: V12, V16 and V20. While doing the line design, it is necessary to dimension the workforce: to know how many workers should be employed on this line. This number should be as small as possible in order to decrease the labor costs. Thus, a workforce planning for this assembly line, minimizing the number of workers, is necessary.

The assembly stations are connected by a unidirectional conveyor. The sequence of the assembly stations to be visited by the semi-finished engines, and the sequences of tasks to be performed at the assembly stations are the same for all engine variants. However, the task processing times depend on the variant of engine. For the same number of workers assigned to a task, some tasks for V20 engines take a longer time than those for V16 engines, and some tasks for V16 engines take more time than those for V12 engines, while some other tasks have the same processing times for all engine variants.

The assembly line is paced, therefore, the sum of task times at any assembly station in each takt must not exceed a given takt time.

To calculate the takt time, the following information given in Table 1 is used.

The annual production volume and shares (ratios) of demand for each engine variant are known, they are given in Table 2.

Taking into account the information that the line is designed to produce 1450 engines per year, the takt time in hours needed to produce one engine is calculated as follows:

$$\text{Takt time} = \frac{230\left[\frac{\text{days}}{\text{year}}\right] * 2\left[\frac{\text{shifts}}{\text{day}}\right] * 8\left[\frac{\text{hours}}{\text{shift}}\right]}{1450\left[\frac{\text{engines}}{\text{year}}\right]} = 2.5\ [\text{hours/engine}]$$

The manufacturer prefers to have evenly distributed engine variants on the line. Due to this requirement, a cyclic sequence of variants to enter the line was designed taking into account the ratios of engine models in the total annual production. These ratios are as follows:

$$\text{Ratio V12} = 75\% = \frac{3}{4} = \frac{15}{20}$$

$$\text{Ratio V16} = 20\% = \frac{1}{5} = \frac{4}{20}$$

$$\text{Ratio V20} = 5\% = \frac{1}{20}$$

Therefore, the following cyclic sequence respecting the criteria of smoothness and ratios was selected: see Fig. 1.

Mass production with push principle, the smoothness criterion for variant releases and given variant ratios can explain this choice made by our industrial partner. The total annual sequence of

Table 1
Available production time.

Days per year	230
Shifts per day	2
Shift duration (h)	8

Table 2
Annual production volume and variant ratios.

Annual volume for all variants (engines/year)	1450
V12 share	75%
V16 share	20%
V20 share	5%

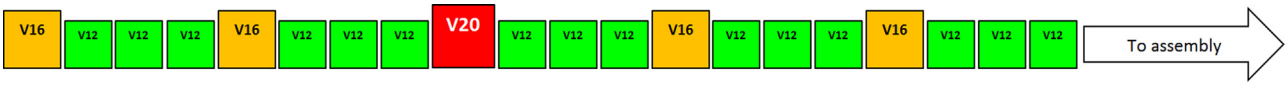

Fig. 1. Cyclic sequence of 20 evenly distributed engine variants.

engine variants is, of course, a repetition of this sequence of 20 regularly distributed engine models.

Without loss of generality we will study only this cyclic sequence. If we need to consider all other situations, we can enumerate all possible sequences respecting the variant ratios, and apply the same approach and model for each of these sequences. Taking into account the constraints on ratios, the number of possible cyclic sequences is not large. For the studied industrial case, in total, there are $\binom{19}{4} = 3876$ possible sequences. Moreover, for each sequence, we can calculate a lower bound on the number of workers as well as an upper bound on the number of workers on the line. Considering these bounds and other constraints, we can remove less promising sequences. For this example, after such an analysis, only 12 sequences were proposed to our industrial partner and the sequence in Fig. 1 was selected.

There are 11 assembly workstations in the line. The first state of the line for the sequence in Fig. 1, i.e. the state at the first takt, is where the 11 first engines of the sequence are loaded on the stations in their order in the sequence (three V12, then one V16, then three V12 and again one V16 and three V12). Before starting the second takt, the V12 engine of the last station will leave the line, all other engines will move to their next station, and the V20 engine (next in the sequence) is loaded on the first station, and so on for next takts. In total, there are 20 states of the line for this sequence. The last state is where the last V16 engine is on the last station of the line. Thus after 20 takts the sequence is completed (the cycle is finished) and it will be repeated again and so on.

For such a line, the processing time of a task for an engine variant is inversely proportional to the number of workers assigned to it. Let task i require p_i time units if it is performed by one worker. Let $p_i(m)$ be the processing time of task i if it is performed by m workers. The relation between time $p_i(m)$ needed to perform task i by m workers, and time p_i needed to perform task i by one worker is as follows:

$$p_i(m) = p_i/m$$

No more than four workers can be assigned to the same task.

Workers are authorized to move from one station to another station when it is needed. If a lack of workforce occurs at a station, then workers from the stations with a surplus of the workforce can move there to help to complete tasks in time. An example of such a situation is illustrated in Fig. 2.

In Fig. 2, at two stations, which are occupied in a certain takt by V20 and V16 engines, two workers are not enough to obtain the sum of processing times satisfying the given takt time. If two workers from stations with a surplus of the workforce move to the station with the V20 engine, and one worker moves to the station with the V16 engine, then the takt time is satisfied.

The distance between any two consequent stations is at most 11 m, and any worker can travel between them in less than 10 s, which is only 0.09% of the takt time. Therefore, we can simplify the problem by assuming that the time needed by any worker to move from one station to another is equal to 0.

In this line, there are also sub-assembly stations. An assembly station can have several associated sub-assembly stations. A sub-assembly module, which is a part of the engine prepared at a sub-assembly station, must be finished in a takt that immediately precedes the takt in which the engine variant will be treated at the corresponding assembly station.

In Fig. 3, the V20 engine arrives at the assembly station A.S.01 and, at the same time, the sub-assembly stations SA.S.2.1 and SA.S.2.2 start to prepare a sub-assembly module for the V20 engine, so that it will be mounted on the engine in the next takt at the assembly station A.S.02.

For the studied assembly line, there are 170 tasks in total to be executed at 11 assembly and 17 sub-assembly stations for one engine. The workers move among all stations (assembly and sub-assembly stations)

The states of sub-assembly stations are completely defined by the state of corresponding assembly stations; thus in our model we do not need to distinguish assembly and sub-assembly stations, we need only to know the state of all stations (assembly and sub-assembly) for a takt, and we will search for an optimal schedule of workers among all 28 stations.

The state of the assembly (and consequently subassembly) stations for a takt are defined by the selected sequence of product variants and the position of the takt in a cycle. Since the annual sequence of engine variants entering the line is a repetition of the sub-sequence presented in Fig. 1 that consists of 20 variants, there are 20 distinct takts which differ with the assignment of engine models to the assembly stations. The problem is to find an assignment of workers to the 170 tasks such that the total number of workers is minimized and the takt time of 2.5 h is satisfied for each of these 20 takts. Since the set of workers should not change from one takt to another one, the number of workers of the line must be set to the maximal number of workers among the 20 different takts. Taking into account our assumption that worker moving times are negligible, the 20 corresponding problems can be solved independently. Thus, we develop a model for only one takt. This model will be used for each of 20 takts of this line.

The initial solution proposed by our industrial partner consisted of 28 workers for 15 takts and 29 workers for the remaining 5 takts. Thus, it was supposed that at most 29 workers are necessary for this line.

3. Overall statement of the problem and related literature

The following formal problem, *denoted problem P*, is studied:

A paced line to assemble different variants of a product is designed. A set of workstations is given, at which tasks required for assembling the product variants are performed. The tasks are already assigned to stations and the assignment and the order of tasks at each station cannot be modified. Product variants enter the line in a given sequence one after another, and all products visit all stations in the same order. At the same station, the same tasks are performed for any type of product. Tasks of each station are performed sequentially.
When a takt is finished, one new product enters the first assembly station of the line, one finished product leaves the last assembly station, and semi-finished products move towards the next assembly stations. The new takt starts.
Each task requires a minimum number of workers but also has an upper limit on the possible number of workers. The task time depends on the number of workers employed for its execution and on the variant of product. The values of processing times are given for each task, number of workers and product variant. If a worker is assigned to a task, he/she is fully occupied by this task from its start till completion time. After

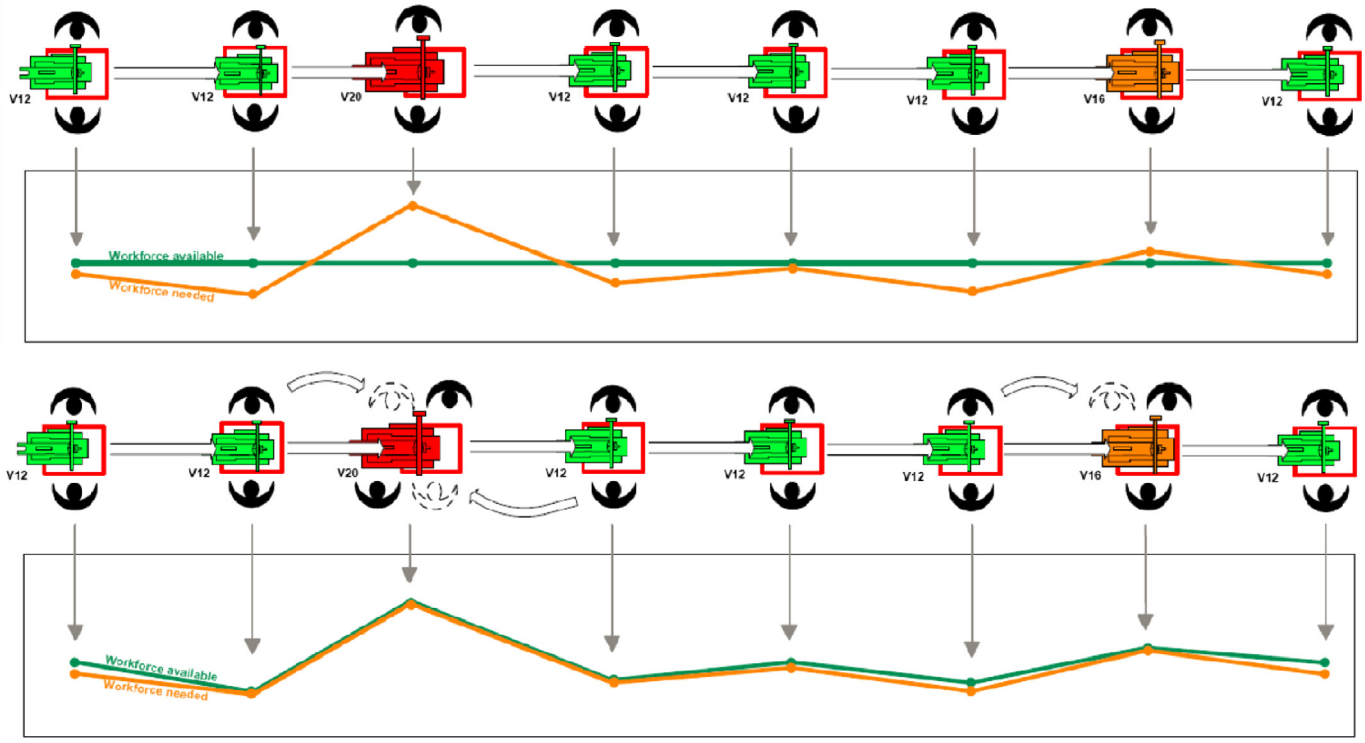

Fig. 2. Adaptation of work capacity to workload by movement of workers along stations.

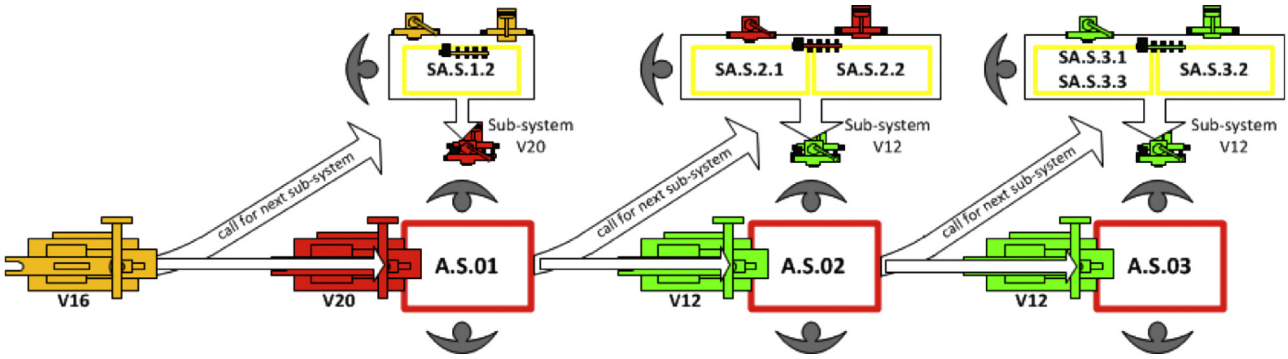

Fig. 3. Relation between sub-assembly and assembly stations.

having completed a task, a worker can move among stations. The travel time is assumed to be zero (it is negligible compared to processing times).

The line takt time is assumed to be given. A takt time is the period of time needed to produce one item of the finished product. It is determined by the required throughput of the assembly line (equal to the inverse of the number of products required per time period). The takt time provides an upper limit on the total processing time for one item at any station.

The sequence of products is given. Thus, for each takt, the assignment of products to the assembly stations is also given and consequently the task times for all stations are known. The problem is to determine for a takt an assignment of workers to tasks, such that the total number of workers is minimized and the line takt time is not exceeded. In other words, the assignment of workers to tasks needs to find a sequence of tasks for each worker for the considered takt. More formally, considering a set W of workers and a set I of tasks, the latter set being partitioned in subsets of ordered tasks which are called stations, the problem is to assign each task to a number of workers.

Problem P needs to be solved for each possible takt. The maximum of the minimal number of workers on all takts will give us the necessary number of workers for the line.

The notion of tasks which can be assigned to some workers working in parallel has already been introduced in scheduling, in the case of tasks assigned to identical parallel processors. Such tasks are named malleable tasks (see, e.g. Blazewicz et al. (2006), Fan et al. (2012), Jansen and Zhang (2012) and Sadykov (2012)). As a scheduling problem, and using the standard notation, problem P can be denoted by Pm | MT, prec, C_{MAX} | m where MT stands for *Malleable Tasks*.

In the literature, the optimization of workforce resources has also been considered in line balancing problems. A general presentation of line design and balancing problems is given in (Dolgui and Proth, 2010) and a recent comprehensive review of literature on assembly line balancing can be found in (Battaïa and Dolgui, 2013). More specifically, assembly line balancing problems with workforce assignment are currently being intensively studied. The most frequently considered criterion is to minimize the line takt time, see Miralles et al. (2008), Blum and Miralles (2011), Moreira and Costa (2013) and Mutlu et al. (2013). Most of the publications

on workforce assignment or workforce planning assume that workers have different skills, which influence task processing times and hence the line productivity; see, for example, Nakade and Ohno (1999), Miralles et al. (2007), Niemi (2009), Araújo et al. (2012), Othman et al. (2012), Fowler et al. (2008) and Costa and Miralles (2009).

The state-of-the-art on workforce planning problems assuming different skills of workers is given in De Bruecker et al. (2015). A workforce assignment problem with a total cost minimization criterion, in which a higher qualified worker can substitute a lower qualified one, but not vice versa, was investigated by Özgüven and Sungur (2013) and Sungur and Yavuz (in press). The authors used integer linear programming models to solve this problem. A specific assembly line balancing problem arising in lines with conventional and disabled workers was studied by Moreira et al. (2015). They proposed several mathematical models and heuristics to minimize the number of workstations, while integrating in the assembly line a number of disabled workers. Borba and Ritt (2014) used an MIP model and a heuristics based on beam search to solve a worker assignment problem with a fixed number of workers and the production rate maximization criterion.

Corominas et al. (2008) study a problem that differs from problem *P* in that it distinguishes permanent and temporary workers, introduces incompatible groups of tasks and a change of stations is not possible. The objective is to minimize the number of temporary workers. A binary linear programming formulation for this problem is proposed and a commercial solver is applied. A heuristic algorithm for the assignment of workers in a lean U-shaped line is developed by Shewchuk (2008). Nakade and Nishiwaki (2008) study a problem of allocating non-identical workers to machines in a U-shaped production line to minimize the takt time, provided that the number of workers is minimized. A constructive heuristic algorithm for a multi-period problem of the workforce assignment in a mixed-model vehicle production assembly line is proposed by Karabak et al. (2011). A mathematical model for scheduling skilled permanent workers and unskilled temporary workers in a mixed-model flow line is developed by Techawiboonwong et al. (2006). In this latter problem, the numbers of skilled and unskilled workers in each time period are the decision variables, and the goal is to minimize wage and hiring/firing costs.

A certain number of publications is dedicated to the effect of the learning, forgetting and aging of the workforce; see for example Grosse and Glock (in press), Grosse et al. (2013), Boenzi et al. (2015) and Hewitt et al. (2015).

In the works of Akagi et al. (1983), Wilson (1986) and Lutz and Davis (1994), it is assumed that the performance of the workers performing the same task is the same, and that the task processing time depends solely on the number of workers assigned to this task. Thus, they made the same assumption as we do for problem *P*.

The assignment of identical workers, just as in problem *P*, was also studied in the literature. Vairaktarakis et al. (2002) developed heuristic algorithms for workforce planning in synchronous production systems with identical well-trained workers. They solved this problem for the cost minimization criterion, including workforce costs and costs associated with the length of the production horizon. Yu et al. (2013) studied the multi-objective problem of assembly line conversion into a pure cell system, for which one of the objectives was to minimize the number of workers, just as in problem *P*. The problem with the same objectives, but for maintenance workforce sizing, was studied by Ighravwe and Oke (2014). The authors formulated a non-linear integer programming model to solve it.

Our research on problem *P* was inspired by a specific workforce assignment problem coming from the automotive industry. The problem with such a formulation has never been studied in literature.

4. MIP formulation of problem *P*

In this section, we suggest a *Mixed-Integer Programming* (*MIP*) model for problem *P*. The model is the same for each of the 20 takts determined by the entering sequence of evenly distributed engine types, but the input data can be different for different takts. The following notation is used:

4.1. Given sets:

- W – set of available workers;
- I – set of tasks;
- L – set of last tasks at all stations;
- O – set of ordered pairs of tasks (i,j) such that task i precedes task j on their station, not necessarily immediately;
- O' – set of ordered pairs of tasks (i,j) such that task i immediately precedes task j on their station;
- D – set of unordered pairs of tasks $\{i,j\}$ such that they are assigned to different stations.

4.2. Indices:

- i or j – a task;
- w – a worker;
- m – number of workers assigned to a task.

4.3. Given parameters:

- C – takt time;
- w_{max} – cardinality of the set W of available workers, $w_{max} = |W|$;
- m_{max} – an upper limit on the number of workers assigned to the same task, $m_{max} \le w_{max}$;
- $p_i(m)$ – processing time of task i, if it is performed by m workers, $i \in I$, $m = 1, \ldots, m_{max}$.

4.4. Decision variables:

- S_i – start time of task i;
- x_{iw} – {1, if i is the first task performed by worker w, 0 otherwise};
- x_{ijw}-{1, if worker w successively performs tasks i and j, and no other task between them, 0 otherwise};
- u_{ij} – {1, if tasks i and j are performed successively by the same any worker, and this worker performs no other task between them, 0 otherwise};
- y_{im} – {1, if task i is performed by m workers, 0 otherwise};
- z_w – {1, if worker w is used, 0 otherwise}.

Let M be a sufficiently large positive number. Our MIP model is as follows:

$$\min \sum_{w=1}^{w_{max}} z_w,$$

subject to

$$S_i + \sum_{m=1}^{m_{max}} p_i(m) \cdot y_{im} \le C, \ \forall i \in L, \tag{1}$$

$$S_j - S_i \ge \sum_{m=1}^{m_{max}} p_i(m) \cdot y_{im}, \ \forall (i,j) \in O', \tag{2}$$

$$S_j - S_i \ge \sum_{m=1}^{m_{max}} p_i(m).y_{im} - M(1-u_{ij}), \ \forall \{i,j\} \in D, \tag{3}$$

$$\sum_{m=1}^{m_{max}} y_{im} = 1, \ \forall i \in I, \tag{4}$$

$$\sum_{m=1}^{m_{max}} m \cdot y_{im} = \sum_{w=1}^{w_{max}} x_{iw} + \sum_{j \in I, j \ne i} \sum_{w=1}^{w_{max}} x_{jiw}, \ \forall \ i \in I, \tag{5}$$

$$x_{ijw} + x_{jiw} \le 1, \ \forall i,j \in I, \ i \ne j, \ w \in W, \tag{6}$$

$$x_{iw} + \sum_{j \in I \setminus \{i\}} x_{jiw} \ge \sum_{j \in I \setminus \{i\}} x_{ijw}, \ \forall \ i \in I, \ w \in W, \tag{7}$$

$$x_{jw} + \sum_{i \in I} x_{ijw} \le z_w, \ \forall \ j \in I, \ w \in W, \tag{8}$$

$$\sum_{i \in I} x_{iw} = z_w, \ \forall w \in W, \tag{9}$$

$$\sum_{w=1}^{w_{max}} x_{ijw} \ge u_{ij}, \ \forall \{i,j\} \in D, \tag{10}$$

$$\sum_{w=1}^{w_{max}} x_{ijw} \le m_{max} \cdot u_{ij}, \ \forall \{i,j\} \in D, \tag{11}$$

$$u_{ij} + u_{ji} \le 1, \ \forall \ i,j \in I, \ i \ne j, \tag{12}$$

$$z_w \ge z_{w+1}, \ \forall w \in W, \ w \ne w_{max}, \tag{13}$$

$$\sum_{j \in I \mid (j,i) \in 0} p_j(m_{max}) \le S_i \le C - \sum_{j \in I \mid (i,j) \in 0} p_j(m_{max}), \ \forall \ i \in I, \tag{14}$$

$$x_{iw} \in \{0,1\}, \ \forall i \in I, \ w \in W, \tag{15}$$

$$x_{ijw} \in \{0,1\}, \ \forall \ i,j \in I, \ w \in W, \tag{16}$$

$$u_{ij} \in \{0,1\}, \ \forall \{i,j\} \in D, \tag{17}$$

$$y_{im} \in \{0,1\}, \ \forall \ i, \ \in I, \ m = 1, \ldots, m_{max}, \tag{18}$$

$$z_w \in \{0,1\}, \ \forall w \in W. \tag{19}$$

The constraints (1) ensure that the takt time limit is not exceeded.

The constraints (2) guarantee that if task i immediately precedes task j at a station, then it completes before or at the start of task j.

The constraints (3) assure that if i and j are decided to be performed by the same worker in the order (i,j) with no other task between them, i.e., $u_{ij} = 1$, then the start time of task i plus its processing time does not exceed the start time of task j.

The constraints (4) ensure that the number of workers assigned to a task is unique.

On the left hand side of the constraints (5), there is the number of workers assigned to task i. On the right hand side, the first sum is the number of workers for which i is the first task. The double sum counts for workers that switch to i immediately from some other tasks. It is required that the values in both sides are equal.

The constraints (6) exclude the case where a worker performs tasks i and j both in the order (i,j) and in the order (j,i).

The constraints (7) require that if a worker performs tasks i and j in the order (i,j) with no other task between them ($\sum_{j \in I, j \ne i} x_{ijw} = 1$ for a worker w), then either i is their first task ($x_{iw} = 1$) or he/she performs some other task immediately before task i ($\sum_{j \in I, j \ne i} x_{jiw} = 1$).

The constraints (8) state that if worker w is not used, i.e., $z_w = 0$, then there are no tasks i and j such that this worker switches to j immediately after i, nor task j which is the first of this worker. If worker w is used, i.e., $z_w = 1$, then there can only be either one task i such that this worker switches to j immediately after i, or one task j which is the first of this worker.

The constraints (9) state that, if a worker is chosen, then his/her first task should be assigned to him/her, and, if a worker is not chosen, then no first task should be assigned to him/her.

The constraints (10) state that, if i and j are decided to be performed by the same worker in the order (i,j) with no other task between them, i.e., $u_{ij} = 1$, then there is at least one worker who can do it.

The constraints (11) require that, if i and j are decided not to be performed by the same worker in the order (i,j) with no other task between them, i.e., $u_{ij} = 0$, then no worker can perform these tasks in this order, with no task between them.

The constraints (12) exclude the case where the tasks i and j are processed both in the order (i,j) and in the order (j,i).

The constraints (13) break the symmetry by requiring that only consecutively numbered workers are used.

An interesting open question is whether the relaxed problem, in which a worker can switch from one task to another, but not necessarily at the end of the former task, has the same optimal value as the original problem. If the answer to this question is positive, then one can solve the problem that determines the numbers of workers for each task, and then solve another problem, which specifies these workers.

5. NP-hardness proof

In this section, we prove that problem P is NP-hard in the strong sense.

Theorem 1. *Problem P is NP-hard in the strong sense even if there is a single task on each station and each task can only be performed by one worker.*

Proof. We will use a reduction from the NP-complete in the strong sense problem 3-Partition; see Garey and Johnson (1979).

3-Partition: Given $3k+1$ positive integer numbers $h_1, \ldots, h_{3k}$ and H satisfying $\sum_{j=1}^{3k} h_j = kH$ and $H/4 < h_j < H/2$, $j = 1, \ldots, 3k$, does there exist a partition of the set $\{1, \ldots, 3k\}$ into subsets $X_1, \ldots, X_k$ such that $\sum_{j \in X_t} h_j = H$ for $t = 1, \ldots, k$?

Given an instance of 3-Partition, we construct an instance of problem P, in which there are $n = 3k$ tasks, k available workers: only one worker can perform any task: $m_{max} = 1$, $3k$ stations, and task j must be performed on station j, $j = 1, \ldots, 3k$. Task processing times are $p_{j1} = h_j$, $j = 1, \ldots, 3k$. The takt time is $C = H$. We will show that there exists a feasible solution for this instance with at most k workers, if and only if, the instance of 3-Partition has a solution. The described reduction is pseudo-polynomial in the input length of 3-Partition.

Let us suppose that there exists a feasible solution of the constructed instance of problem P. Note that all k workers must be used, because otherwise at least one worker will need to perform at least four tasks, whose total duration will exceed the takt time H. For the same reason, each worker must perform exactly three tasks. Let X_t denote the set of tasks performed by worker t, $t=1,\ldots,k$. Due to the solution feasibility, relations $\sum_{j\in X_t} h_j \le H$, $t=1,\ldots,k$, must hold. They imply $\sum_{j\in X_t} h_j = H$, $t=1,\ldots,k$, as it is required to prove the part "only if" of our claim.

Conversely, if a collection of sets $X_1,\ldots,X_k$ is a solution of 3-Partition, then assign tasks of the set X_t to the worker t, who will perform them in any order, $t=1,\ldots,k$. For this solution, all tasks will complete by $H=C$. Hence, it is a solution to the constructed instance of problem P. □

The above proof implies that problem P is hard in strong sense, thus an efficient exact algorithm to solve P is unlikely to exist. This means that regardless of a given available calculation time, there is always a size of problem's instance for which the problem cannot be solved optimally. Therefore, in the next section, we introduce several heuristics to be able to treat problem P of any practical size.

6. Heuristics

The first heuristic is simple. It determines the number of workers for each station. They perform every task of their station and do not move to another station. This heuristic gives the same solution as the solution proposed by our industrial partner.

Let s be the number of stations and let I_q be the given set of tasks of station q, $q=1,\ldots,s$. Let m_q denote the number of workers assigned to station q, which has to be determined, $m_q=1,\ldots,m_{max}$. The first heuristic is as follows:

6.1. ***Heuristic*** *Same − Station:*

- **Step 1.** Calculate $m_q = \min\{m \mid \sum_{j\in I_q} p_j(m) \le C\}$, $q=1,\ldots,s$.

  Note that, if $\sum_{j\in N} p_j(m_{max}) \le C$, then the bisection search can be employed to find the above minimum in polynomial time for any non-increasing function $p_j(m)$. In each of $O(\log_2 m_{max})$ iterations of the bisection search, the relation $\sum_{j\in I} p_j(m) \le C$ needs to be verified for a trial value $m \in \{1,\ldots,m_{max}\}$.
- **Step 2**. Output a solution, in which m_q consecutively indexed workers perform tasks required for station q and these tasks only, $q=1,\ldots,s$.

  If $m_q > m_{max}$ for a q, $1\le q\le s$, then the solution found is unfeasible. Otherwise, it is feasible with the total number of workers $W^{(1)} = \sum_{q=1}^{s} m_q$.

This algorithm can be applied with any functions $p_j(m)$. For the case $p_j(m) = p_j/m$, with this algorithm we obtain m_q equal to $\sum_{j\in I_q} p_j/C$.

Our second heuristic, denoted as *Sequential − Stations*, is also applied for $m_q \le m_{max}$, $q=1,2,\ldots,s$. The heuristic considers stations in a certain sequence. Let the sequence be $1,2,\ldots,s$. The heuristic proposes to assign the same arbitrary m_{max} workers to the stations $1,2,\ldots,q_{max}$, $q_{max}\le s-1$, so that the workers serve these stations in the indicated order from time zero until the total processing time of the last task of station q_{max} does not exceed the takt time C. Then the assigned workers and the stations $1,\ldots,q_{max}$ are removed from the problem input, and the process is repeated. If $q_{max}=s$, then $m_0 < m_{max}$ workers can be assigned to the last stations.

6.2. ***Heuristic*** *Sequential − Stations:*

- **Step 1**. Calculate $m_q = \min\{m \mid \sum_{j\in I_q} p_j(m) \le C\}$, $q=1,2,\ldots,s$. If $m_q > m_{max}$ for a q, $1\le q\le s$, then stop: no feasible solution exists.
- **Step 2**. Re-number stations $1,2,\ldots,s$ in a certain order, for example, such that the sub-assembly stations of station 1 go first, then station 1, then the sub-assembly stations of station 2, station 2, and so on. Set $a=1$. Initialize the total number of assigned workers $W^{(2)}=0$.
- **Step 3**. Determine $q_{max} = \max_{a\le q\le s}\{q \mid \sum_{h=a}^{q}\sum_{j\in I_h} p_j(m_{max}) \le C\}$.

  If $q_{max}=s$, then go to Step 4. If $q_{max}\le s-1$, then perform the following computations. Assign arbitrary m_{max} workers from the set W of available workers to each task of the stations $a, a+1,\ldots,q_{max}$ so that they serve these stations in the indicated order. Remove the assigned workers from the set W. Re-set $a := q_{max}+1$ and $W^{(2)} := W^{(2)} + m_{max}$. Repeat Step 3.
- **Step 4**. Determine $m_0 = \min_{1\le m\le m_{max}}\{m \mid \sum_{h=a}^{s}\sum_{j\in I_h} p_j(m) \le C\}$.

Assign arbitrary m_0 workers to each task of the stations $a, a+1,\ldots,s$ so that they serve these stations in the indicated order. Re-set $W^{(2)} = W^{(2)} + m_0$. Output the final solution and its value $W^{(2)}$.

The third heuristic, denoted as *Sequential-Stations-Random*, differs from the heuristic *Sequential-Stations* in that the sequence of stations is generated at random.

6.3. ***Heuristic*** *Sequential-One-Traveling-Worker:*

Let s be the number of stations and let I_q be the given set of tasks required at station q, $q=1,\ldots,s$. Let m_q denote the number of workers assigned to station q, which has to be determined, $q=1,\ldots,m_{max}$, where i or j is a task, $i,\ j \in I$.

If the number of workers m_q, performing tasks of station q with a given takt time C, is constant, then we can evaluate the minimum number of workers m_q required to perform the given set of tasks I_q at station q for a given tact time C:

$$m_q = \min\{m \mid \sum_{j\in I_q} p_j(m) \le C\},\ q=1,\ldots,s.$$

Now we consider the case when a part of the tasks of set I_q is performed by m_q workers and a part of the tasks of set I_q is performed by m_q-1 workers.

Let $I^-_{q,k}$ be first k tasks of station q and $I^+_{q,k} = I_q \setminus I^-_{q,k}$.

We can calculate:

$$k_q = \min\left\{ k \mid \sum_{j\in I^-_{q,k}} p_j(m_q) + \sum_{j\in I^+_{q,k}} p_j(m_q-1) \le C \right\}$$

Note that $I^-_{q,k} \ne \varnothing$, and $I^+_{q,k}$ can be an empty set.

If $I^+_{q,k} \ne \varnothing$, then tasks of set I^-_{q,k_q} are performed by m_q workers and tasks of set I^+_{q,k_q} are performed by m_q-1 workers. In this situation, one worker at station q is free starting from the time:

$$S_q = \sum_{j\in I^-_{q,k}} p_j(m_q)$$

This worker can perform tasks of set $I_{q+1,r}$ at station $q+1$. We assume that all tasks assigned to a particular station are denoted sequentially and first task performed at a station has index 1, so:

$$I_{q+1,r} = \{\, j\in I_{q+1},\ j\ge r\} \tag{20}$$

If all the workers initially assigned to the station $q+1$ remain at this station until the end of takt time, we can calculate r using the

following formula:

$$r = \min\left\{ r \mid \sum_{j \in I_{q+1} j < r} p_j(m_{q+1}) \geq S_q \right\}$$

However we should note that this formula does not take into account that, due to the set $I^+_{q+1,k}$, one of the workers at station $q+1$ does not necessarily perform all tasks from set $I^* = \{ j \mid j \in I_{q+1},\ j < r \}$, because of his/her movement to station $q+2$. We will provide the exact value for r after clarifying the partition of all tasks of station $q+1$ into subsets that are performed by different numbers of workers.

Now we need to calculate the minimum number of workers m_{q+1} required to perform a given set of tasks I_{q+1} at station $q+1$, taking into account that a worker from station q performs tasks $I_{q+1,r}$. To make these calculations, we consider the following equalities:

$$I_q = I^-_{q,k} \cup I^+_{q,k}$$

$$I_q = (I_q \backslash I_{q,r}) \cup I_{q,r}$$

$$I_q \backslash I_{q,r} = (I^-_{q,k} \backslash I_{q,r}) \cup (I^+_{q,k} \backslash I_{q,r})$$

$$I_{q,r} = (I_{q,r} \backslash I^-_{q,k}) \cup (I_{q,r} \backslash I^+_{q,k})$$

Therefore:

$$I_q = (I^-_{q,k} \backslash I_{q,r}) \cup (I^+_{q,k} \backslash I_{q,r}) \cup (I_{q,r} \backslash I^-_{q,k}) \cup (I_{q,r} \backslash I^+_{q,k})$$

The number of workers for each subset is shown in Table 3.

Figs. 4 and 5 demonstrate the subsets defined above.

Since we have already defined how many workers are performing tasks for each subset (see Table 3), we can provide an exact value for r:

$$r = \min\left\{ r \mid \sum_{j \in I^-_{q+1,k} \backslash I_{q+1,r}} p_j(m_{q+1}) + \sum_{j \in I^+_{q+1,k} \backslash I_{q+1,r}} p_j(m_{q+1}-1) \geq S_q \right\} \tag{21}$$

Note that set $I_{q,r}$ defined by formula (21) contains the set $I_{q,r}$ defined by formula (20).

One can think that we can use the following formula to determine m_{q+1}:

$$m_{q+1} = \min\left\{ m \mid k_{q+1} = \min\left\{ k \mid \sum_{j \in I^-_{q+1,k} \backslash I_{q+1,r}} p_j(m_{q+1}) + \sum_{j \in I^+_{q+1,k} \backslash I_{q+1,r}} p_j(m_{q+1}-1) + \sum_{j \in I_{q+1,r} \backslash I^-_{q+1,k}} p_j(m_{q+1}) + \sum_{j \in I_{q+1,r} \backslash I^+_{q+1,k}} p_j(m_{q+1}+1) \leq C \right\} \right\}. \tag{22}$$

However, this formula does not guarantee that there will be no overflow for one worker and underflow for another one (see an explanation below). We need to add constraints for each worker and then combine them. As the result we will obtain the following:

$$m_{q+1} = \min\left\{ m \mid k_{q+1} = \min\left\{ k \mid \sum_{j \in I^-_{q+1,k} \backslash I_{q+1,r}} p_j(m_{q+1}) + \sum_{j \in I^+_{q+1,k} \backslash I_{q+1,r}} p_j(m_{q+1}-1) \leq S_q,\ \sum_{j \in I_{q+1,r} \backslash I^-_{q+1,k}} p_j(m_{q+1}) + \sum_{j \in I_{q+1,r} \backslash I^+_{q+1,k}} p_j(m_{q+1}+1) \leq C - S_q \right\} \right\} \tag{23}$$

Table 3
Number of workers for each subset of I_q.

$I^-_{q,k} \backslash I_{q,r}$	m
$I^+_{q,k} \backslash I_{q,r}$	$m-1$
$I_{q,r} \backslash I^-_{q,k}$	m
$I_{q,r} \backslash I^+_{q,k}$	$m+1$

Thus, one worker at station $q+1$ does not perform tasks of set $(I^+_{q+1,k} \backslash I_{q+1,r}) \cup (I_{q+1,r} \backslash I^-_{q+1,k}) = I^+_{q+1,k}$. This worker can perform tasks of set $I_{q+2,r}$ at station $q+2$ starting from the time S_{q+1}:

$$S_{q+1} = \sum_{j \in I^-_{q+1,k} \backslash I_{q+1,r}} p_j(m_{q+1}) + \sum_{j \in I_{q+1,r} \backslash I^+_{q+1,k}} p_j(m_{q+1}+1) + \Delta_{q+1} \tag{24}$$

$$\Delta_{q+1} = \begin{cases} 0, & r \geq k \\ S_q - \sum_{j \in I^-_{q+1,k} \backslash I_{q+1,r}} p_j(m_{q+1}) + \sum_{j \in I^+_{q+1,k} \backslash I_{q+1,r}} p_j(m_{q+1}-1), & r < k \end{cases}$$

Note that $S_{q+1} \geq \sum_{j \in I^-_{q+1,k}} p_j(m_{q+1})$, due to possible idle time between tasks.

The term Δ_{q+1} appears in equality (24) because the time S_q is evaluated for tasks of station q but the sets $I^+_{q+1,k}$, $I^-_{q+1,k}$ are defined for tasks of station $q+1$. Therefore, the idle time shown in Fig. 5 appears between tasks of set $I^-_{q+1,k} \backslash I_{q+1,r}$ and tasks of set $I_{q+1,r} \backslash I^+_{q+1,k}$. For this reason, in order to calculate m_{q+1} we use formula (23) instead of formula (22).

Note that we can define S_{q+1} in another way:

$$S_{q+1} = \begin{cases} \sum_{j \in I^-_{q+1,k} \backslash I_{q+1,r}} p_j(m_{q+1}), & r \geq k \\ S_q + \sum_{j \in I_{q+1,r} \backslash I^+_{q+1,k}} p_j(m_{q+1}+1), & r < k \end{cases}$$

6.4. ***Heuristic*** *Min-Idle-One-Traveling-Worker:*

In the previous heuristic we used sequentially arranged stations. However, we can improve this heuristic by choosing the next station for a worker to move to; it will be the station which will have the minimal total idle time after adding this supplementary worker. Note that we should only take into account idle time but not the free time (the rest of the time at the start of moving, i.e. the difference between the end of takt and the moment when the move starts) of the worker who is going to move to the next station.

6.5. ***Heuristic*** *Random-One-Traveling-Worker:*

Another way of choosing the sequence of stations is at random.

6.6. ***Heuristic*** *Additional-Workers:*

This heuristic is as follows (we assume that workers performing tasks can move from one station to another).

Step 1. For every station q, $q = 1, \ldots, s$, we evaluate the number of workers m_q so that

$$\sum_{j \in I_q} p_j(m_q+1) < C \leq \sum_{j \in I_q} p_j(m_q), \quad q = 1, \ldots, s,$$

and the remainder time (slack time):

$$r_q = \sum_{j \in I_q} p_j(m_q) - C$$

And we assign m_q workers to each station q.

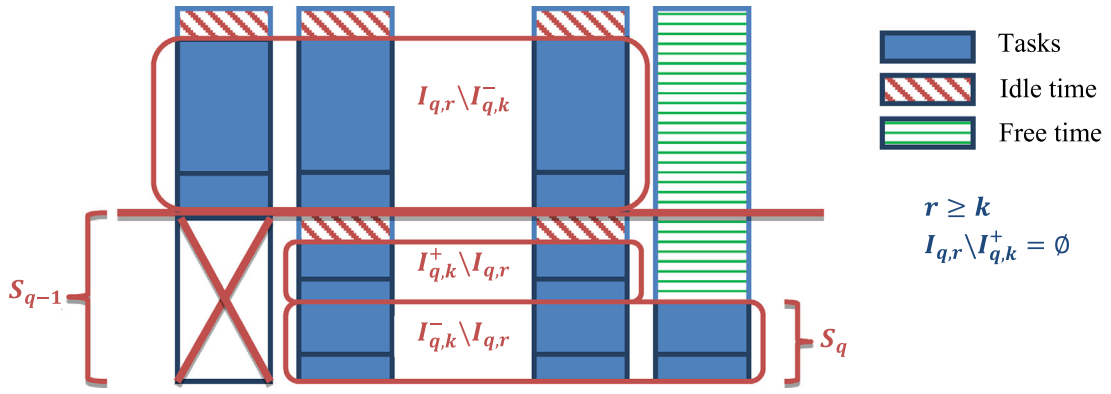

Fig. 4. Graphic representation of I_q subsets in case if $r \geq k$ and $I_{q,r}\backslash I^+_{q,k} = \varnothing$.

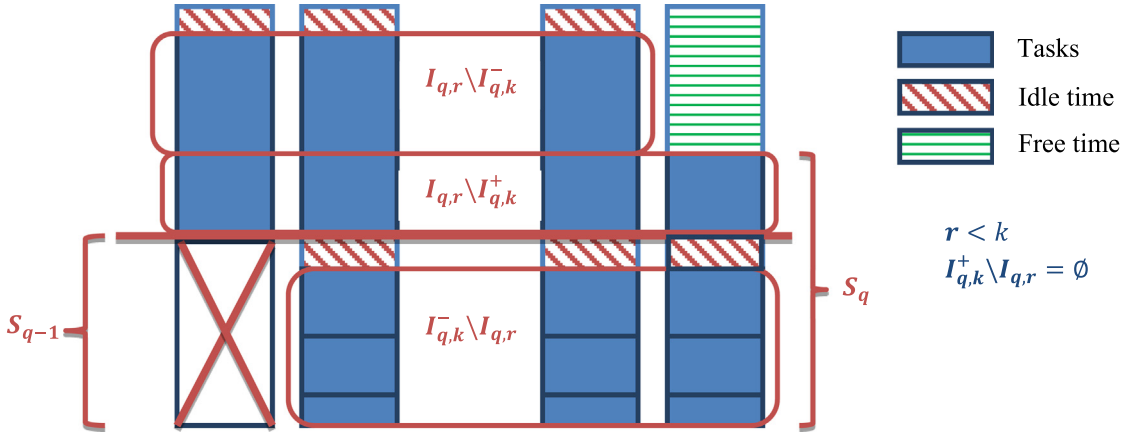

Fig. 5. Graphic representation of I_q subsets in case if $r < k$ and $I^+_{q,r}\backslash I_{q,k} = \varnothing$.

Step 2. Now, the problem is to find a minimal set of additional workers $\{w_d, 1 \leq d \leq D\}$ to perform the set of tasks I. This problem is equivalent to the problem of placement of intervals with length r_q, $q = 1, \ldots, s$, without intersections in the minimal number of intervals with length C.

For station $q = 1$, let b_{q-1} be equal to 0 and let the index of additional worker d be equal to 1. Find minimal set of tasks $I_{q,r}$ for station q such that they are performed after time b_{q-1} and their runtime is greater than or equal to r_q. In order to do this, we calculate:

$$a_q = \min\left\{a \mid \sum_{j \in I_q, j \leq a} p_j(m_q) \geq b_{q-1}\right\};$$

$$b_q = \min\left\{b \mid \sum_{j \in I_q, a_q \leq j < b} p_j(m_q + 1) \geq r_q\right\}.$$

Note that we can also use the following formula for b_q:

$$b_q = \min\left\{b \mid \sum_{j \in I_q, j < a_q} p_j(m_q) + \sum_{j \in I_q, a_q \leq j < b} p_j(m_q + 1) + \sum_{j \in I_q, j \geq b} p_j(m_q) < C\right\}$$

Then, $I_{q,r} = \{j \in I_q,\ a_q \leq j < b_q\}$. If $I_{q,r}$ exists, we add the set $I_{q,r}$ to tasks that are performed by the worker with number d, then go to next station $q+1$ and find a minimal set of tasks $I_{q+1,r}$. If the set $I_{q,r}$ does not exist, we take a new additional worker w_{d+1}, assign $b_{q-1} = 0$ and find a minimal set of tasks $I_{q,r}$ for station q.

6.7. ***Heuristic*** *Additional-Workers-Improved:*

We can also consider the following modification of the heuristic proposed above. For every station q, $q = 1, \ldots, s$, we evaluate $b_q = \min\{b \mid \sum_{j \in I_q, a_q \leq j < b} p_j(m_q + 1) \geq r_q\}$, where a_q does not depend on b_{q-1}; here a_q is selected randomly or so that $b_q - a_q$ is minimized. We get intervals $[a_q, b_q]$, $q = 1, \ldots, s$. Then, we solve the problem of placement of intervals $[a_q, b_q]$ without intersections in the minimal number of intervals $[0, C]$.

7. Computer experiments

In order to compare proposed heuristics they were implemented using C# programming language. Fig. 6 demonstrates a screenshot of the created software application.

The experiments were run on PC with Intel Core i7-3520M CPU with 4 Gb of RAM under MS Windows 7, 64 bit. All the heuristics take just a few seconds to provide a solution, which is very fast (Fig. 7).

The time limit for solving problem P with the MIP model for each of the 20 takts was set to 30 min in a commercial solver. However, we could not obtain an exact solution, due to the exponentially increasing time needed to find it. Taking into account that problem P must be solved 20 times, i.e. for each takt corresponding to a certain disposition of products on the line, this gives 10 h in total to solve the whole problem. Of course at the design stage, we have enough time, but we believe that psychologically this is inacceptable for a decision maker to wait so long for a solution, which is not even optimal. Moreover, we made computer

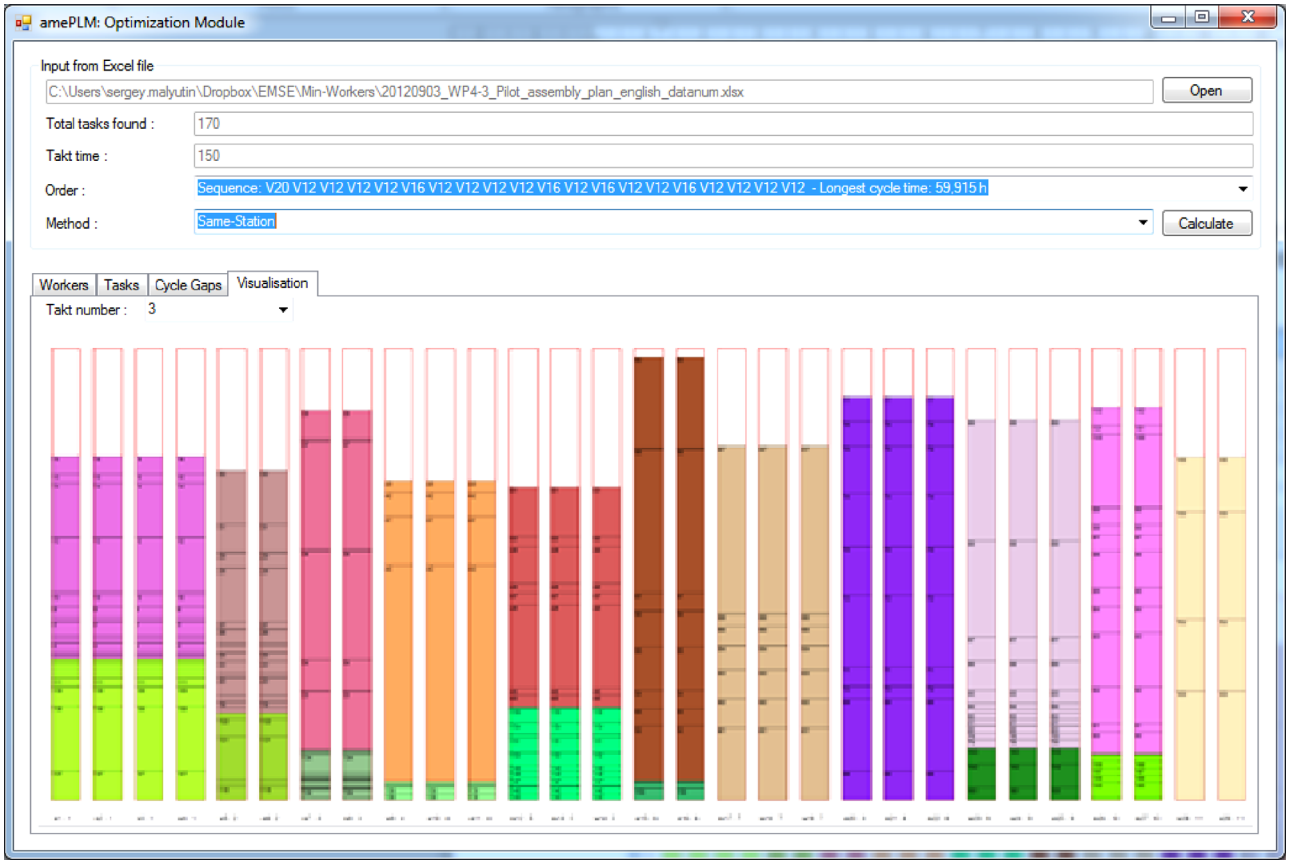

Fig. 6. Screenshot of the created application.

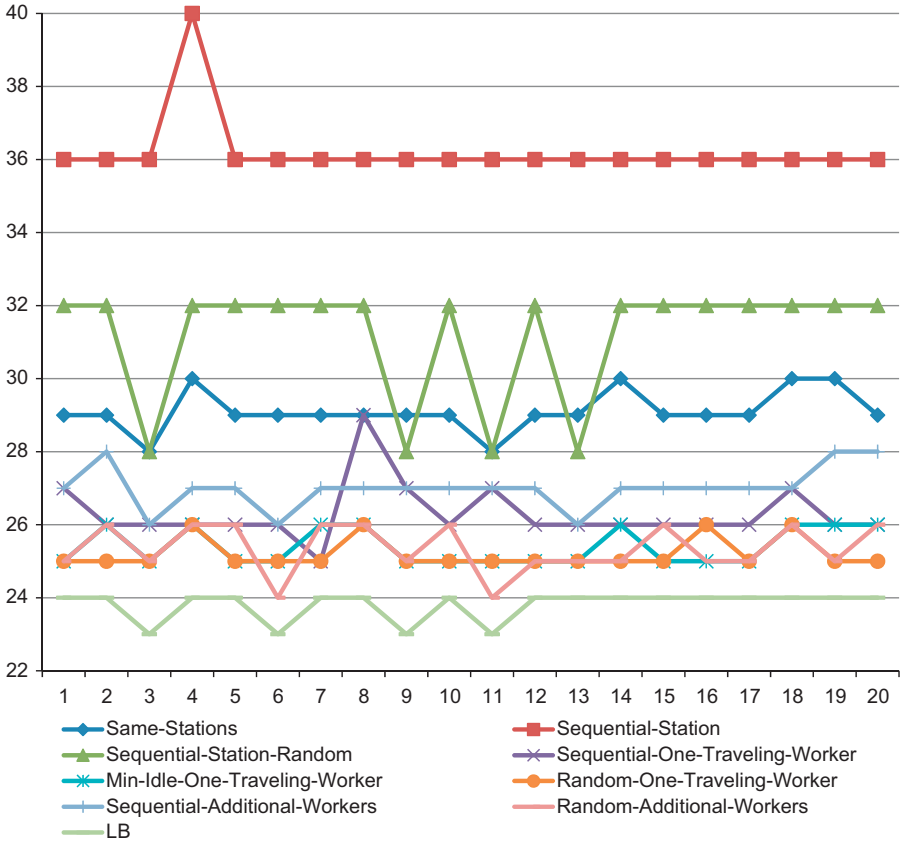

Fig. 7. Comparison of the heuristics' performance (number of workers for each takt).

Table 4
Objective values and relative errors of the obtained solutions.

Takt	LB	Same-Stations		Sequential-Station		Sequential-Station-Random		Sequential-One-Traveling-Worker		Min-Idle-One-Traveling-Worker		Random-One-Traveling-Worker		Sequential-Additional-Workers		Random-Additional-Workers	
		Workers	Gap (%)	Workers	Gap (%)	Workers	Gap (%)	Workers	Gap (%)	Workers	Gap (%)	Workers	Gap (%)	Workers	Gap (%)	Workers	Gap (%)
1	24	29	20.8	36	50.0	32	33.3	27	12.5	27	12.5	25	4.2	27	12.5	25	4.2
2	24	29	20.8	36	50.0	32	33.3	26	8.3	26	8.3	25	4.2	28	16.7	26	8.3
3	23	28	21.7	36	56.5	28	21.7	26	13.0	26	13.0	25	8.7	26	13.0	25	8.7
4	24	30	25.0	40	66.7	32	33.3	26	8.3	26	8.3	26	8.3	27	12.5	26	8.3
5	24	29	20.8	36	50.0	32	33.3	26	8.3	26	8.3	25	4.2	27	12.5	26	8.3
6	23	29	26.1	36	56.5	32	39.1	26	13.0	26	13.0	25	8.7	26	13.0	24	4.3
7	24	29	20.8	36	50.0	32	33.3	25	4.2	25	4.2	25	4.2	27	12.5	26	8.3
8	24	29	20.8	36	50.0	32	33.3	29	20.8	29	20.8	26	8.3	27	12.5	26	8.3
9	23	29	26.1	36	56.5	28	21.7	27	17.4	27	17.4	25	8.7	27	17.4	25	8.7
10	24	29	20.8	36	50.0	32	33.3	26	8.3	26	8.3	25	4.2	27	12.5	26	8.3
11	23	28	21.7	36	56.5	28	21.7	27	17.4	27	17.4	25	8.7	27	17.4	24	4.3
12	24	29	20.8	36	50.0	32	33.3	26	8.3	26	8.3	25	4.2	27	12.5	25	4.2
13	24	29	20.8	36	50.0	28	16.7	26	8.3	26	8.3	25	4.2	26	8.3	25	4.2
14	24	30	25.0	36	50.0	32	33.3	26	8.3	26	8.3	25	4.2	27	12.5	25	4.2
15	24	29	20.8	36	50.0	32	33.3	26	8.3	26	8.3	25	4.2	27	12.5	26	8.3
16	24	29	20.8	36	50.0	32	33.3	26	8.3	26	8.3	26	8.3	27	12.5	25	4.2
17	24	29	20.8	36	50.0	32	33.3	26	8.3	26	8.3	25	4.2	27	12.5	25	4.2
18	24	30	25.0	36	50.0	32	33.3	27	12.5	27	12.5	26	8.3	27	12.5	26	8.3
19	24	30	25.0	36	50.0	32	33.3	26	8.3	26	8.3	25	4.2	28	16.7	25	4.2
20	24	29	20.8	36	50.0	32	33.3	26	8.3	26	8.3	25	4.2	28	16.7	26	8.3
Max	24	30	25.0	40	66.7	32	33.3	29	20.8	29	20.8	26	8.3	28	16.7	26	8.3

experiments for an industrial case in which the sequence of 20 takts is considered with only 3 types of engines and 170 tasks assigned to 28 workstations. In other cases, sequences may be longer with more different engines, tasks and workstations, and thus the size of model for a takt and the number of takts to be considered increase quickly, which results in a larger number of necessary runs where each run is drastically longer, therefore with an exponential increase of overall computational time.

Table 4 contains the objective values and relative errors of the solutions obtained by the heuristics.

The relative error %Gap of a solution is calculated by using the lower bound $LB = \sum_{j \in I} p_j / C$ and the number of workers obtained by the corresponding heuristic:

$$\%\text{Gap} = \frac{\text{Number of Workers} - \text{LB}}{\text{LB}} \times 100\%$$

The workload distribution is unbalanced – some workers have a higher percentage of idle time than others. In order to smooth the workload, we propose to rank workers in takt q (which is just finished) in the non-increasing order of their accumulated idle times. For the next takt $q+1$, the workers are ranked in the non-increasing order of their workload. Then, for this takt $q+1$, the worker with the highest workload will be replaced (to perform his/her tasks) by the worker with the highest accumulated idle time in the takt q. For example, let worker w_1 have the highest accumulated idle time up to and including the current takt q, and let worker w_2 have the highest workload in the next takt $q+1$ according to the obtained solution. Then, in the takt $q+1$, worker w_1 will perform the tasks of worker w_2.

8. Conclusions

In this paper, we have studied the problem of workforce resources optimization for manual mixed-model assembly lines. We developed models and algorithms to assign identical workers to tasks minimizing the total number of workers required. This research is motivated by a real industrial case from the automotive industry but the interest of this study is more general, since the situation of multiple workers working in parallel and able to move among workstations could occur in various industries using mixed-model assembly lines.

The studied assembly line is manual designed to produce several variants of a product. The assignment of tasks to workstations is given. Each task processing time depends only on the engine variant and the number of workers performing this task. The problem consists in assigning a minimal possible number of workers to tasks so that the imposed takt time is not exceeded.

First, we proposed a MIP model for each production takt which corresponds to a disposition of engine models on the assembly line. Then, we have proved that the problem is NP-hard in the strong sense. Finally, we suggested eight constructive heuristics, three of which are randomized. Computer experiments proved the efficiency of the proposed heuristics, both in terms of solution quality and computational time.

The analysis of the obtained solutions revealed a high disproportion of the workload between the assigned workers. In order to cope with this problem, we also proposed a dynamic procedure, which reassigns the workers: the workers with the highest accumulated idle time in the current takt will execute tasks of workers with the highest workload in the next takt.

Acknowledgments

This work was supported by FoF-ICT-2011.7.4 Collaborative Project (grant 285171) “amePLM” of the European Commission, EU within the Seventh Framework Program (2007–2015).

References

Akagi, F., Osaki, H., Kikuchi, S., 1983. A method for assembly line balancing with more than one worker in each station. Int. J. Prod. Res. 21 (5), 755–770.
Araújo, F.F.B., Costa, A.M., Miralles, C., 2012. Two extensions for the ALWABP: parallel stations and collaborative approach. Int. J. Prod. Econ. 140 (1), 483–495.
Battaïa, O., Dolgui, A., 2013. A taxonomy of line balancing problems and their solution approaches. Int. J. Prod. Econ. 142 (2), 259–277.
Blazewicz, J., Kovalyov, M.Y., Machowiak, M., Trystram, D., Weglarz, J., 2006. Preemptable malleable task scheduling problem. IEEE Trans. Comput. 55 (4), 486–490.

Blum, C., Miralles, C., 2011. On solving the assembly line worker assignment and balancing problem via beam search. Comput. Oper. Res. 38 (1), 328–329.
Boenzi, F., Mossa, G., Mummolo, G., Romano, V.A., 2015. Workforce aging in production systems: modeling and performance evaluation. Procedia Eng. 100, 1108–1115.
Borba, L., Ritt, M., 2014. A heuristic and a branch-and-bound algorithm for the assembly line worker assignment and balancing problem. Comput. Oper. Res. 45, 87–96.
Costa, A.M., Miralles, C., 2009. Job rotation in assembly lines employing disabled workers. Int. J. Prod. Econ. 120 (2), 625–632.
Corominas, A., Pastor, R., Plans, J., 2008. Balancing assembly line with skilled and unskilled workers. Omega 36 (6), 1126–1132.
De Bruecker, P., Van den Bergh, J., Beliën, J., Demeulemeester, E., 2015. Workforce planning incorporating skills: state of the art. Eur. J. Oper. Res. 243 (1), 1–16.
Dolgui, A., Proth, J-M., 2010. Supply Chain Engineering: Useful Methods and Techniques. Springer, London, UK.
Fan, L., Zhang, F., Wang, G., Liu, Z., 2012. An effective approximation algorithm for the Malleable Parallel Task Scheduling problem. J. Parallel Distrib. Comput. 72, 693–704.
Fowler, J.W., Wirojanagud, P., Gel, E.S., 2008. Heuristics for workforce planning with worker differences. Eur. J. Oper. Res. 190 (3), 724–740.
Garey, M.R., Johnson, D.S., 1979. Computers and Intractability: A Guide to the NP-Completeness. W. H. Freeman & Company, New York, USA.
Grosse, E.H., Glock, C.H., 2015. The effect of worker learning on manual order picking processes. Int. J. Prod. Econ. http://dx.doi.org/10.1016/j.ijpe.2014.12.018, in press.
Grosse, E.H., Glock, C.H., Jaber, M.Y., 2013. The effect of worker learning and forgetting on storage reassignment decisions in order picking systems. Comput. Ind. Eng. 66 (4), 653–662.
Hewitt, M., Chacosky, A., Grasman, S.E., Thomas, B.W., 2015. Integer programming techniques for solving non-linear workforce planning models with learning. Eur. J. Oper. Res. 241 (3), 942–950.
Ighravwe, D.E., Oke, S.A., 2014. A non-zero integer non-linear programming model for maintenance workforce sizing. Int J. Prod. Econ. 150, 204–214.
Jansen, K., Zhang, H., 2012. Scheduling malleable tasks with precedence constraints. J. Comput. Syst. Sci. 78, 245–259.
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, pp. 483–490.
Lutz, C.M., Davis, K.R., 1994. Development of operator assignment schedules: a DSS approach. Omega 22 (1), 57–67.
Miralles, C., García-Sabater, J.P., Andrés, C., Cardós, M., 2007. Advantages of assembly lines in sheltered work centres for disabled. A case study. Int J. Prod. Econ. 110 (1–2), 187–197.
Miralles, C., García-Sabater, J.P., Andrés, C., Cardós, M., 2008. Branch and bound procedures for solving the assembly line worker assignment and balancing problem: application to sheltered work centres for disabled. Discret. Appl. Math. 156, 357–367.
Moreira, M.C.O., Costa, A.M., 2013. Hybrid heuristics for planning job rotation schedules in assembly lines with heterogeneous workers. Int J. Prod. Econ. 141 (2), 552–560.
Moreira, M.C.O., Miralles, C., Cost, A.M., 2015. Model and heuristics for the assembly line worker integration and balancing problem. Comput. Oper. Res. 54, 64–73.
Mutlu, O., Polat, O., Supciller, A.A., 2013. An iterative genetic algorithm for the assembly line worker assignment and balancing problem of type-II. Comput. Oper. Res. 40 (1), 418–426.
Nakade, K., Nishiwaki, R., 2008. Optimal allocation of heterogeneous workers in a U-shaped production line. Comput. Ind. Eng. 54 (3), 432–440.
Nakade, K., Ohno, K., 1999. An optimal worker allocation problem for a U-shaped production line. Int. J. Prod. Econ. 60–61 (1), 353–358.
Niemi, E., 2009. Worker allocation in make-to-order assembly cells. Robot. Comput.-Integr. Manuf. 25 (6), 932–936.
Othman, M., Bhuiyan, N., Gouw, G.J., 2012. Integrating workers' differences into workforce planning. Comput. Ind. Eng. 63 (4), 1096–1106.
Özgüven, C., Sungur, B., 2013. Integer programming models for hierarchical workforce scheduling problems including excess off-days and idle labour times. Appl. Math. Model. 37 (22), 9117–9131.
Sadykov, R., 2012. A dominant class of schedules for malleable jobs in the problem to minimize the total weighted completion time. Comput. Oper. Res. 39, 1265–1270.
Shewchuk, J.P., 2008. Worker allocation in lean U-shaped production lines. Int. J. Prod. Res. 46 (13), 3485–3502.
Sungur, B., Yavuz, Y., 2015. Assembly line balancing with hierarchical worker assignment. J. Manuf. Syst. http://dx.doi.org/10.1016/j.jmsy.2014.08.004, in press.
Techawiboonwong, A., Yenradee, P., Das, S.K., 2006. A master scheduling model with skilled and unskilled temporary workers. Int J. Prod. Econ. 103 (2), 798–809.
Vairaktarakis, G.L., Cai, X., Lee, C.-Y., 2002. Workforce planning in synchronous production systems. Eur. J. Oper. Res. 136 (1), 551–572.
Venkatesh, J., 2008. Evaluation of performance measures for representing operational objectives of a mixed model assembly line balancing problem. Int. J. Prod. Res. 46, 6367–6388.
Wilson, J.M., 1986. Formulation of a problem involving assembly lines with multiple manning of work stations. Int. J. Prod. Res. 24 (1), 59–63.
Yu, Y., Tang, J., Sun, W., Yin, Y., Kaku, I., 2013. Reducing worker(s) by converting assembly line into a pure cell system. Int J. Prod. Econ. 145 (2), 799–806.