INSEEC Research
PARIS • BORDEAUX • LYON • ALPES-SAVOIE
MONACO • LONDON • CHICAGO

Working Paper
n°: 2016-100-01

Minimizing the number of stations and station activation costs for a production line

Sergey Kovalev [a], Xavier Delorme [b], Alexandre Dolgui [c], Ammar Oulamara [d]

a INSEEC Business School, 25 rue de l'Université, 69007 Lyon, France
b Ecole des Mines de Saint-Etienne, LIMOS UMR CNRS 6158, 158 cours Fauriel, 42023, Saint-Etienne Cedex 2, France
c Ecole des Mines de Nantes, IRCCYN, UMR CNRS 6597, La Chantrerie, 4 rue Alfred Kastler – B.P. 20722, F-44307 Nantes Cedex 3, France
c University of Lorraine, Ile de Saulcy, 57045 Metz, France

May 2015

An ulterior version of this article is to appear in Computers and Operations Research
It can be purchased at: http://www.sciencedirect.com/science/article/pii/S0305054816302556

Minimizing the number of stations and station activation costs for a production line

Sergey Kovalev[1], Xavier Delorme[2], Alexandre Dolgui[2] and Ammar Oulamara[3]

[1]*INSEEC Business School ECE Lyon, 19 place Tolozan, 69001 Lyon, France, e-mail: smkovalev@yahoo.com*

[2] *LIMOS UMR CNRS 6158, École des Mines de Saint-Étienne, 158, cours Fauriel, 42023 Saint-Étienne Cedex 2, France, e-mail: {delorme,dolgui}@emse.fr*

[3]*University of Lorraine, Ile de Saulcy, 57045 Metz, France, e-mail: ammar.oulamara@loria.fr*

Abstract

A bicriterion problem of a multi-product production line design has recently been studied, in which the primary criterion is the minimization of the number of stations and the secondary criterion is the minimization of the station activation costs. Sequential execution of operations on the same station was assumed. We extend these earlier studies to another widely encountered type of operations processing at a station – simultaneous. Computational complexity for various special cases of this new problem are established. Heuristic algorithms, integer linear programming formulations, and computer experiments are presented. Instances of practical dimension, with 40 to 80 operations, are solved in an hour on a conventional computer.

Keywords: Line balancing; Scheduling; Bicriteria optimization; Computational complexity; Integer linear programming; Heuristics.

1 Problem definition and relevant literature

A problem of line design with activation costs is considered. This is a new line balancing problem especially important for industries where activation costs cannot be neglected. Activation costs are different from setup costs because they are associated with each part and not with a part type (lot). A station is activated for part processing if at least one operation on this part will be processed at this station.

A formal definition of the problem under study can be given as follows. A paced production line consisting of several stations has to be configured to produce parts of f types. Parts move along the stations in the same direction in a given sequence. Let $F = \{1, \ldots, v, \ldots, f\}$ be the set of part types. Each part of type $v \in F$ requires each operation of a given set N_v to be executed exactly once on the line. Operations of the set N_v are called *type v operations*. The same operation can be performed on parts of different types, thus, different sets N_v can contain common operations. Let $N := \cup_{v=1}^{f} N_v = \{1, \ldots, n\}$ denote the superset of all the required operations, and let T_i denote the set of types of the operation $i \in N$, i.e., $T_i = \{v \mid i \in N_v,\ v \in F\}$.

Operations of the same type assigned to the same station are performed simultaneously. Examples of simultaneous execution of operations in the mechanical industry are given by Dolgui et al. [17], Belmokhtar et al. [5] and Battaia and Dolgui [3], and in the automotive industry by Falkenauer [21]. Re-design, i.e., re-assignment of operations when switching from the production of one type part to another, is not allowed.

Binary *precedence relations* are given on the superset N. If operation $i \in N$ precedes operation $j \in N$, then j cannot be assigned to the station of i or any preceding station. Precedence relations are transitive and irreflexive. They are represented by a directed acyclic graph $G = G(N, A)$, in which there is an arc $(i, j) \in A$ if and only if i precedes j. Precedence relations characterize the technological process. They can be defined separately for each part type, but since re-design is not allowed, they must be consistent and can therefore be given on the superset N.

Plural *exclusion relations* are defined on the set N. They are represented by a collection E of subsets $E' \subset N$ such that all operations of E' cannot be assigned to the same station, but any proper subset of E' can be assigned to the same station. These relations prevent simultaneous utilization of tools which are in conflict due to their physical characteristics. Assume, without loss of generality, that there are no two sets in E containing one another. For example, if there are two exclusion sets $\{1, 2, 3\}$ and $\{1, 3\}$, then $\{1, 2, 3\}$ can be removed from the input since the

fact that operations 1 and 3 will never be assigned to the same station implies that 1, 2 and 3 will do the same. Again, exclusion relations can be defined separately for each part type, but since re-design is not allowed, they must be consistent and can therefore be given on the superset N.

Each operation i has its *size* s_i which is the number of tools (workers) required to perform this operation. The total size of all operations, i.e., the total number of tools assigned to the same station should not exceed the same given station capacity r.

Since each tool is associated with an *elementary operation*, each operation i can be viewed as a block of elementary operations to be executed simultaneously on the same station, with s_i as the cardinality of this block and N as a set of blocks. Therefore, operations with non-unit sizes are similar to the *inclusion sets* such that all operations of the same inclusion set must be assigned to the same station. Inclusion sets are used for elementary operations whose precision can be lost if they are assigned to different stations, see Dolgui et al. [18] for an inclusion sets formulation for a transfer line balancing problem.

Exclusion and inclusion relations refer to what is called *zoning constraints* in line balancing, see Akpinar and Mirac Bayhan [1] and Falkenauer [21]. Zoning constraints are called positive and negative if they oblige and prohibit, respectively, the assignment of specific operations to the same station. In our case, exclusion relations correspond to negative zoning constraints while inclusion relations represented by operation sizes refer to positive zoning constraints.

A station is activated for part processing if at least one operation on this part will be processed at this station. An activation cost a_v is associated with part type v, $v = 1, \ldots, f$. These activation costs are related to the additional resources required to start and finish part processing at a station, see Borgia et al. [6].

Let x_v denote the number of stations activated for any single part of type v, $v = 1, \ldots, f$. A decision has to be made about the total number of stations, k, and an assignment of the operations to the stations $1, \ldots, k$. The primary criterion is to minimize the total number of stations, which is a classic criterion for the line balancing problems, and the secondary criterion is to minimize the total activation cost, $a_1x_1 + \cdots + a_fx_f$. We denote the problem with just the primary criterion as $P(prec, excl, s_i \mid k)$ and the problem with the two lexicographically ordered criteria as $P(prec, excl, s_i \mid k, cost)$.

Note that the two criteria are not redundant in the sense that minimizing the total activation cost is not sufficient to minimize the number of stations and vice versa. Let (k^*, c^*) denote the

lexicographical minimum of the objective function values for the problem $P(prec, excl, s_i \mid k, cost)$. Thus, k^* is the global minimum of the number of stations, and c^* is the minimum cost, provided that the number of stations is k^*.

For an illustration of the effect of station activation costs, assume that there are three stations, each having capacity 2, and two types of parts. Any part of type 1 requires unit size operations 1, 2 and 3 and any part of type 2 requires unit size operations 4, 5 and 6. Assume that the other constraints are not effective. Two feasible assignments of the six operations are given in Fig. 1.

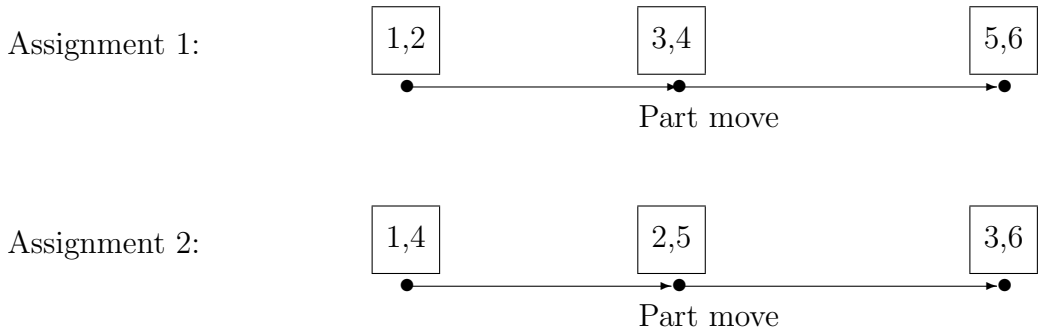

Figure 1: Two assignments of operations

Consider assignment 1. When a part of type 1 is processed, station 1 performs operations 1 and 2, and station 2 performs operation 3. Station 3 is not activated. This contributes $2a_1$ to the total station activation cost. When a part of type 2 is processed, station 2 performs operation 4, and station 3 performs operations 5 and 6. Station 1 is not activated. This contributes $2a_2$ to the total station activation cost, which is equal to $2a_1 + 2a_2$ for the assignment 1. Now consider assignment 2. When a part of type 1 is processed, station 1 performs operation 1, station 2 performs operation 2 and station 3 performs operation 3. This contributes $3a_1$ to the total station activation cost. When a part of type 2 is processed, station 2 performs operation 4, station 2 performs operation 5 and station 3 performs operation 6. This contributes $3a_2$ to the total station activation cost, which is equal to $3a_1 + 3a_2$ for the assignment 2.

Kovalev et al. [30] studied a special case of the problem $P(prec, excl, s_i \mid k, cost)$ with unit size operations and no precedence or exclusion relation, which we denote as $P(s_i = 1 \mid k, cost)$. They suggested a solution procedure with running time depending solely on the number of part types f. This procedure combines the enumeration of different assignments of part types with the solution of a system of linear inequalities. Dolgui et al. [16] investigated a problem similar to $P(prec, excl, s_i \mid k, cost)$, in which operations of the same type assigned to the same station

are performed sequentially and there are no exclusion relations. The cycle time constraints were explicitly present in the formulation of this problem because of the sequential execution of operations at the same station. NP-hardness proofs, polynomial time algorithms for special cases, heuristics and integer linear programming formulations for the general problem were suggested in [16]. These results cannot be applied to the problem $P(prec, excl, s_i \mid k, cost)$ because the difference between sequential and simultaneous processing makes the solution structures and feasible domains of these problems substantially different. The fact that the type of processing drastically changes the mathematical model and affects solution methods is widely observed in the line balancing problems.

Problem $P(prec, excl, s_i \mid k)$ is related to multi-product line balancing problems, which are studied by Sawik [36], Kabir and Tabucanon [29], Bukchin and Rabinowitch [11], Vilarinho and Simaria [39], Andres et al. [2], Yuan et al. [41], among others. Surveys of line balancing research are given by Erel and Sarin [20], Rekiek et al. [35], Boysen et al. [9], Dolgui and Proth [19], Boysen et al. [10] and Battaia and Dolgui [4]. Results on multi-objective line balancing problems can be found in Gamberini et al. [24], McMullen and Tarasewich [32, 33], Suwannarongsri and Puangdownreong [37]. Setup time and setup cost models are well known in scheduling and lot-sizing research, see, for example, Potts and Kovalyov [34].

The rest of the paper is organized as follows. In Section 2, we focus on the problem with the sole primary criterion. The computational complexity of this problem, which is NP-hard in the strong sense in the general case, is analyzed. Two greedy randomized heuristics and an integer linear programming (ILP) formulation are suggested. In Section 3, the lexicographic bi-criteria problem is studied. Again the computational complexity of this problem is analyzed, and two greedy randomized heuristics as well as an ILP formulation are suggested. Computer experiments with the suggested heuristics and ILP formulations are given in Section 4. A solution procedure for the bi-criteria problem, which dichotomically enumerates the number of stations, is proposed in Section 5. The paper concludes with a summary of the results.

2 Minimizing the number of stations

In this section, the single criterion problem $P(prec, excl, s_i \mid k)$ to minimize the number of stations is studied. The simplest case with unit size operations and no exclusion or precedence relations is easily solvable: $k^* = \lceil \frac{n}{r} \rceil$. In the following subsections, more complex cases are considered.

2.1 Given precedence relations, no exclusion relation and unit sizes

The special case in this subsection is denoted as $P(prec, s_i = 1 \mid k)$. We will show that it is equivalent to the scheduling problem $Pm \mid p_i = 1, prec \mid C_{\max}$.

Problem $Pm \mid p_i = 1, prec \mid C_{\max}$: There are n unit-time jobs to be scheduled on m identical parallel machines. Precedence relations are given on the set of jobs such that if job i precedes job j then j cannot be processed in the unit-time processing interval of i or any preceding unit-time interval. No preemption of job processing is allowed and no two jobs can be processed by the same machine simultaneously. The objective is to assign jobs to the unit-time intervals on the machines so that the precedence relations are satisfied and the completion time of the latest job, $C_{\max}$, is minimized. Denote the minimum $C_{\max}$ value as $C^*_{\max}$.

For problem $P(prec, s_i = 1 \mid k)$, let us interpret operations as jobs, the capacity bound r as the number of machines m, and stations $1, 2, \ldots$ as unit-time intervals $[0, 1], [1, 2], \ldots$ in the problem $Pm \mid p_i = 1, prec \mid C_{\max}$. It can be seen that any feasible solution of problem $P(prec, s_i = 1 \mid k)$ with the number of stations k can be represented as a feasible solution of the corresponding scheduling problem with the objective value $C_{\max} = k$ and vice versa. Therefore, the two problems are equivalent and $k^* = C^*_{\max}$.

If precedence relations are arbitrary, then problem $Pm \mid p_i = 1, prec \mid C_{\max}$ is NP-hard in the strong sense for a variable number of machines m (Ullman [38]), and its computational complexity status is open if m is a constant greater than two (Brucker and Knust [12]). Problem $Pm \mid p_i = 1, prec \mid C_{\max}$ is solvable in $O(n)$ time if the precedence graph, G, is a collection of in-trees (Hu [28]) or a collection of out-trees (Davida and Linton [14]), and it is solvable in $O(n^2)$ time, if G is arbitrary and $m = 2$ (the best known algorithm is by Gabow [23]).

Thus, problem $P(prec, s_i = 1 \mid k)$ is NP-hard in the strong sense for variable capacity bound r and arbitrary precedence graph G. It is polynomially solvable if G is a collection of in-trees or a collection of out-trees, or if $r = 2$.

The chain-like and tree-like precedence relations are typical for machining and assembly operations. Below we give our interpretation of the algorithm by Hu [28] for the problem with the in-tree precedence relations. There, the words "operation" and "vertex" are used interchangeably. We substitute notation $prec$ for $in - tree$ or $out - tree$ in the problems with the in-tree (respectively, out-tree) precedence relations. Let the stations be numbered $t = 1, 2, \ldots$

Algorithm HLF (Highest Level First) for the problem $P(in - tree, s_i = 1 \mid k)$:

Step 1 (Preparation of the input data) With each vertex $j \in N$, store its direct predecessors, their number, and its direct successor. Calculate the *level* (number of direct and indirect successors) l_j of each vertex $j \in N$ by scanning vertices from the roots to the leafs: the roots get level zero, their direct predecessors get level one, and so on. With each vertex $j \in N$, store its level l_j. Let $l_{\max} = \max\{l_j | j \in N\}$. For each $l = 0, 1, \ldots, l_{\max}$, form the list L_l of vertices which have level l and have no predecessor. Form the list of vertices $L = (L_{l_{max}}, L_{l_{\max}-1}, \ldots, L_0)$ which have no predecessors and appear in the non-increasing order of their levels. Some lists L_l in L can be empty. Set $t = 1$.

Step 2 (Operation assignment) Assign the first operation (with the highest level) of the list L to station t. If all the operations of the set N are assigned, then stop - an optimal solution is constructed and $k^* = t$. Otherwise, update list L by removing its first operation. Decrease by one the number of predecessors of the direct successor of the assigned operation. If the number of predecessors of a vertex of level l has become zero, store this vertex in an auxiliary list Z_l. If all the operations of the list L are assigned or if all the r positions of the station t are filled, then update list L by re-setting $L_l := (L_l, Z_l)$ for each auxiliary list Z_l. Re-set $t := t + 1$ and repeat Step 2. ■

Algorithm HLF can be implemented to run in $O(n)$ time. The same algorithm can be used to solve problem $P(out - tree, s_i = 1 \mid k)$ if we reverse precedence constraints and consider an optimal solution for the problem $P(in - tree, s_i = 1 \mid k)$ obtained from the last station to the first.

2.2 Arbitrary operation sizes and no exclusion or precedence relation

Denote the special case in this subsection as $P(s_i \mid k)$. The problem $P(s_i \mid k)$ can be interpreted as the well-known bin packing problem, in which there are n items of sizes $s_1, \ldots, s_n$ to be packed into the minimum number of bins, each having capacity r. The minimum number of stations k^* is equal to the minimum number of the required bins.

The bin packing problem is NP-hard in the strong sense. It is solvable in $O(n^{K^L})$ time if there is at most K distinct item sizes and at most L items can fit in one bin; see, for example, Fernandez de la Vega and Lueker [22]. If r is a constant, then K and L are constants as well, and the bin packing problem is polynomially solvable. It can be reduced to solving n parallel machine scheduling problems $Pm \mid d_j = r \mid \cdot$, $m = 1, \ldots, n$. In the problem $Pm \mid d_j = r \mid \cdot$,

there are n non-preemptive jobs with processing times $s_1, \ldots, s_n$ to be scheduled on m identical parallel machines, and the question is whether all the jobs can be completed by the common deadline r. We have $k^* = m^*$ where m^* is the minimum m for which the question in the problem $Pm \mid d_j = r \mid \cdot$ has the affirmative answer. The problem $Pm \mid d_j = r \mid \cdot$ can be solved in $O(nmr^{m-1})$ time by a dynamic programming algorithm. Hence, the problem $P(s_i \mid k)$ can be solved in $O(n^2 r^{n-1} \log_2 n)$ time by a bisection search over the range $1, \ldots, n$ of values m.

From the above discussion we deduce that the problem $P(s_i \mid k)$ is NP-hard in the strong sense. It is polynomially solvable if the number of operations is a constant or if the station capacity r is a constant.

2.3 Given exclusion relations, no precedence relation and unit sizes

Denote the special case in this subsection as $P(excl, s_i = 1 \mid k)$.

Theorem 1 *The problem $P(excl, s_i = 1 \mid k)$ is NP-hard in the strong sense if $|E'| = 2$ for each $E' \in E$.*

Proof: We use a reduction from the problem Exact Cover by 3-Sets (X3C); see Garey and Johnson [25]: Given a family $A = \{A_1, \ldots, A_a\}$ of 3-element subsets of the set $B = \{1, \ldots, 3b\}$, does A contain an exact cover of B, i.e., a subfamily $S \subseteq A$ such that each $i \in B$ belongs to exactly one 3-element set in S? Assume $a > b$ because otherwise there exists a trivial solution.

Given an instance of X3C, construct the following instance of the decision version of the problem $P(excl, s_i = 1 \mid k)$. Define the number of operations $n = (b+1)(a-b+1)$ and the station capacity $r = b + 1$. The set of operations includes:

- a single *X-operation*,

- b number of the same *Y_i-operations*, $i = 1, \ldots, a - b$,

- *A_i-operations* associated with the subsets A_i, $i = 1, \ldots, a$.

All operations are of the same type. The exclusion relations are given as follows. The X-operation excludes any Y_i-operation, i.e., $\{X, j\} \in E$ for every Y_i-operation j, $i = 1, \ldots, a - b$. Any Y_i-operation excludes any Y_q-operation, i.e., $\{j, l\} \in E$ for every Y_i-operation j and every Y_q-operation l, $i \neq q$. The A_i-operation and the A_q-operation exclude each other if subsets A_i and A_q share the same element, i.e., if $A_i \cap A_q \neq \phi$, $i \neq q$.

We will show that for the constructed instance, a solution to X3C exists *if and only if* there exists a feasible solution to this instance of the problem $P(excl, s_i = 1 \mid k)$ with at most $a - b + 1$ stations. It is obvious that the instance construction is pseudo-polynomial.

Part "if". Assume that there exists a feasible solution to the constructed instance of the problem $P(excl, s_i = 1 \mid k)$ with at most $a - b + 1$ stations. Observe that there are $a - b + 1$ mutually exclusive operations: the X-operation and one Y_i-operation for each $i = 1, \ldots, a - b$. Hence, there must be exactly $a - b + 1$ stations. Without loss of generality assume that an Y_i-operation is assigned to station i, $i = 1, \ldots, a - b$, and the X-operation is assigned to the station $a - b + 1$. Since any Y_i-operation excludes any Y_q-operation for $i \neq q$, and there are $a - b + 1$ stations, all the Y_i-operations must be assigned to station i, $i = 1, \ldots, a - b$. Therefore, the A_i-operation, $i \in \{1, \ldots, a\}$, can be assigned to one of the remaining b positions of the station $a-b+1$ or to the single position of any of the stations $1, \ldots, a-b$. The structure of such a solution is given in Table 1 where columns represent stations and symbols "$\bullet$" represent A_i-operations.

Table 1: Structure of a feasible solution with $a - b + 1$ stations.

Y_1	Y_2	$\bullet$	$\cdots$	Y_{a-b}	X
$\bullet$	Y_2	Y_3	$\cdots$	Y_{a-b}	$\bullet$
Y_1	$\bullet$	Y_3	$\cdots$	$\bullet$	$\bullet$
Y_1	Y_2	Y_3	$\cdots$	Y_{a-b}	$\bullet$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
Y_1	Y_2	Y_3	$\cdots$	Y_{a-b}	$\bullet$

The b number of A_i-operations at the station $a-b+1$ must be mutually non-exclusive, which means that no two subsets A_i and A_q corresponding to these operations must share the same element. We deduce that the subsets A_i corresponding to the A_i-operations assigned to station $a - b + 1$ constitute an exact cover of B, as required for the proof of the part "if".

Part "only if". If there exists a solution S to the problem X3C, then construct a solution to the problem $P(excl, s_i = 1 \mid k)$ with the structure given in Table 1 and assign the A_i-operation to the station $a - b + 1$ if and only if $A_i \in S$. Such a solution is feasible and the number of stations is equal to $a - b + 1$, as required for the proof of the part "only if". ∎

2.4 ILP formulation for the general case of $P(prec, excl, s_i \mid k)$

Let us now formulate this problem as an Integer Linear Program (ILP). In this model, we assume that an upper bound $k^0 \geq k^*$ on the number of stations is given, i.e. there are k^0 stations to

be opened or not. In practice, the number k^0 can be obtained by heuristics (see section 2.5). A station is called *open* if at least one operation is assigned to it.

Introduce 0-1 variables x_{ij} such that $x_{ij} = 1$ if and only if operation i is assigned to station j, and 0-1 variables y_j such that $y_j = 1$ if and only if station j is open. Recall that the directed graph $G = G(N, A)$ represents the precedence constraints.

Calculate the following lower bound LB_1 on the minimum number of stations: $LB_1 = \max\left\{l_{\max}, \left\lceil \frac{\sum_{i=1}^{n} s_i}{r} \right\rceil\right\}$, where $l_{\max}$ is the number of vertices in the longest chain of G. The problem $P(prec, excl, s_i \mid k)$ can be re-formulated as the following ILP problem, which we denote as $Min\#$.

Problem $Min\#$:

$$\min \sum_{j=1}^{k^0} y_j, \tag{1}$$

subject to

$$\sum_{j=1}^{k^0} y_j \geq LB_1, \tag{2}$$

$$\sum_{i=1}^{j} y_i \geq j y_j, \quad j = 2, \ldots, k^0, \tag{3}$$

$$\sum_{i=1}^{n} x_{ij} \geq y_j, \quad j = 1, \ldots, k^0, \tag{4}$$

$$\sum_{j=1}^{k^0} x_{ij} = 1, \quad i = 1, \ldots, n, \tag{5}$$

$$\sum_{i=1}^{n} s_i x_{ij} \leq r y_j, \quad j = 1, \ldots, k^0, \tag{6}$$

$$\sum_{i \in E'} x_{ij} \leq |E'| - 1, \ \forall E' \in E, \ j = 1, \ldots, k^0, \tag{7}$$

$$\sum_{j=1}^{h} x_{bj} + \sum_{j=h}^{k^0} x_{aj} \leq 1, \ \forall (a, b) \in A, \ h = 1, \ldots, k^0, \tag{8}$$

$$x_{ij}, y_j \in \{0, 1\}, \quad i = 1, \ldots, n, \ j = 1, \ldots, k^0. \tag{9}$$

In the problem $Min\#$, there are $k^0 + nk^0$ variables and $n + k^0(3 + |E| + |A|)$ constraints without the 0-1 constraints. Barring the exclusion constraints (7), this problem is similar to the simple assembly line balancing problem, SALBP-1, see, for example, Boysen and Fliedner [8].

Constraint (2) incorporates the lower bound LB_1. Constraints (3) ensure that the stations are opened consecutively in the increasing order of their indices and (4) that at least one operation

is assigned to an open station. Note that constraints (2)-(4) are not necessary for the model to be correct but they accelerate the solution process when using an ILP solver. Constraints (5) ensure that each operation is assigned to a station. Constraints (6) guarantee that the capacity of each station is not exceeded and that any operation can be assigned only to an open station. Constraints (7) prevent operations of the same exclusion set to be all assigned to the same station. Constraints (8) together with (5) guarantee that the station index of a is strictly smaller than that of b for $(a, b) \in A$, which is the requirement of the precedence constraints.

The minimum number of stations can be calculated as $k^* = \sum_{j=1}^{k^0} y_j^*$ where y_j^* are optimal y_j values in the problem $Min\#$.

2.5 Heuristics for the general case of $P(prec, excl, s_i \mid k)$

As indicated, the problem considered in this paper is NP-Hard. In addition, the ILP proposed requires an upper bound on the number of stations. Thus, in this section we suggest two randomized greedy heuristics to solve the problem $P(prec, excl, s_i \mid k)$ approximately. Let the stations be numbered $t = 1, 2, \ldots$

Heuristic $GreedyNumber1$ for the problem $P(prec, excl, s_i \mid k)$:

Step 1 (Elimination of exclusion relations) If there is no exclusion set ($E = \phi$), then set $t = 1$ and go to Step 2. Otherwise, select a set $E' \in E$. Randomly select a pair of operations in E' and introduce an arbitrary precedence relation between them. Note that if this precedence relation is satisfied, then the exclusion relation given by E' is satisfied as well. Update the precedence graph $G = G(N, A)$. Update E by removing E' and any other exclusion set which contains a pair of operations with a directed path between them in the new graph G. Repeat Step 1.

Step 2 (Operation assignment by solving *Subset Sum* **problem)** Calculate the set N^+ of vertices of the precedence graph G which have no predecessor. Assign to station t operations of a subset $X \in N^+$, which is selected by solving the following *Subset Sum* problem:

$$\max_{X \in N^+} \sum_{i \in X} s_i, \text{ subject to} \sum_{i \in X} s_i \leq r.$$

Update graph G by removing vertices of the set X and their outgoing arcs. If G is empty, then stop: a feasible solution is constructed. Otherwise, re-set $t := t + 1$ and repeat Step 2.

■

Iterative solving of the *Subset Sum* problem is proven to be an efficient solution approach for the bin packing problems; see, for example, Caprara and Pferschy [13] and Haouari and Serairi [26]. To solve the *Subset Sum* problem, we use the standard dynamic programming algorithm, which requires $O(|N^+|r)$ time.

Heuristic $GreedyNumber1$ can be independently run several times to increase the probability to obtain a feasible solution with a small number of stations.

Heuristic $GreedyNumber2$ differs from $GreedyNumber1$ in that in Step 2, instead of solving the *Subset Sum* problem, the subset $X \in N^+$ such that $\sum_{i \in X} s_i \leq r$ is generated by selecting its elements by random.

3 Minimizing the total activation cost

In this section, we establish computational complexity and present heuristics and an ILP formulation for the problem $P(prec, excl, s_i \mid k, cost)$.

3.1 Computational complexity

Note that any special case of the problem $P(prec, excl, s_i \mid k, cost)$ is not easier than the same special case of the problem $P(prec, excl, s_i \mid k)$. Therefore, NP-hardness results for the problem $P(prec, excl, s_i \mid k)$ apply for the problem $P(prec, excl, s_i \mid k, cost)$.

Kovalev et al. [30] presented a solution algorithm for the problem with unit size operations and no precedence or exclusion relation, which we denote as $P(s_i = 1 \mid k, cost)$. The run time of this algorithm depends solely on the number f of part types. Therefore, the problem $P(s_i = 1 \mid k, cost)$ is polynomially solvable for a constant number of part types. We now show that the problem $P(s_i = 1 \mid k, cost)$ is NP-hard in the strong sense if f is variable (a part of the problem instance) even if all the cost coefficients are unit.

Theorem 2 *The problem $P(s_i = 1 \mid k, cost)$ is NP-hard in the strong sense even if $a_v = 1$, $v = 1, \ldots, f$.*

Proof: We use a reduction from the strongly NP-complete problem 3-Partition; see Garey and Johnson [25]: Given $3q+1$ positive integer numbers $b_1, \ldots, b_{3q}$ and B such that $B/4 < b_i < B/2$, $i = 1, \ldots, 3q$, and $\sum_{i=1}^{3q} b_i = qB$, is there a partition of the set $\{1, \ldots, 3q\}$ into q disjoint sets $X_1, \ldots, X_q$ such that $\sum_{i \in X_l} b_i = B$ for $l = 1, \ldots, q$?

Given an instance of 3-PARTITION, construct the following instance of the problem $P(s_i = 1 \mid k, cost)$. Define the number of operations $n = qB$ and the station capacity $r = B$. There are $f = 3q$ types of operations, each operation has a single type and a unit size, and there are b_i operations of type i, $i = 1, \ldots, 3q$. Thus, there are qB operations in total. We show that for the constructed instance, a solution to 3-PARTITION exists *if and only if* there exists a feasible solution to the constructed instance of the problem $P(s_i = 1 \mid k, cost)$ with q stations and total activation cost equal to $3q$. It is obvious that the construction is pseudo-polynomial.

Part "if". Assume that there exists a feasible solution to the constructed instance of the problem $P(s_i = 1 \mid k, cost)$ with at most q stations and at most $3q$ activation. It is easy to see that the minimum number of stations is equal to q. Since there are $3q$ part types, operations of the same type must be assigned to the same station. Let X_h denote the set of part types assigned to station h. We must have $\sum_{i \in X_h} b_i \leq r = B$, $h = 1, \ldots, q$, which, together with the equality $\sum_{h=1}^{q} \sum_{i \in X_h} b_i = \sum_{i=1}^{3q} b_i = qB$ implies $\sum_{i \in X_h} b_i = B$, $h = 1, \ldots, q$, as required for the proof of the part "if".

Part "only if". If there exists a solution $X_1, \ldots, X_q$ to the problem 3-PARTITION, then construct a solution to the problem $P(s_i = 1 \mid k, cost)$ in which all the operations of the types from the set X_h are assigned to the station h, $h = 1, \ldots, q$. This solution has q stations and the total cost of $3q$, as required for the proof of the part "only if". ∎

3.2 ILP formulation

We now present an ILP formulation for the general case of the problem $P(prec, excl, s_i \mid k, cost)$. Assume that the minimum number of stations k^* has been found. Introduce 0-1 variables z_{vj} such that $z_{vj} = 1$ if and only if at least one operation of type v is assigned to station j. As before for the problem $Min\#$, x_{ij} are 0-1 variables such that $x_{ij} = 1$ if and only if operation i is assigned to station j.

Calculate the following lower bound LB_2 on the minimum total cost: $LB_2 = \sum_{v=1}^{f} a_v \max\left\{l_{\max}^{(v)}, \left\lceil \frac{\sum_{i \in N_v} s_i}{r} \right\rceil \right\}$, where $l_{\max}^{(v)}$ is the maximum number of type v operations in any chain of G. The problem $P(prec, excl, s_i \mid k, cost)$ can be re-formulated as the following ILP problem, which is denoted as $MinAct$.

Problem $MinAct$:

$$\min \quad \sum_{j=1}^{k^*} \sum_{v=1}^{f} a_v z_{vj}, \tag{10}$$

subject to

$$\sum_{j=1}^{k^*}\sum_{v=1}^{f} a_v z_{vj} \geq LB_2, \tag{11}$$

$$\sum_{i\in N_v} x_{ij} \geq z_{vj}, \quad v=1,\ldots,f,\ j=1,\ldots,k^*, \tag{12}$$

$$\sum_{i=1}^{n} s_i x_{ij} \leq r, \quad j=1,\ldots,k^*, \tag{13}$$

$$\sum_{i\in N_v} x_{ij} \leq |N_v| z_{vj},\ v=1,\ldots,f,\ j=1,\ldots,k^*, \tag{14}$$

$$\sum_{j=1}^{k^*} x_{ij} = 1, \quad i=1,\ldots,n, \tag{15}$$

$$\sum_{i\in E'} x_{ij} \leq |E'|-1,\ \forall E'\in E,\ j=1,\ldots,k^*, \tag{16}$$

$$\sum_{j=1}^{h} x_{bj} + \sum_{j=h}^{k^*} x_{aj} \leq 1,\ \forall (a,b)\in A,\ h=1,\ldots,k^*, \tag{17}$$

$$x_{ij}, z_{vj} \in \{0,1\}, \quad i=1,\ldots,n,\ j=1,\ldots,k^*,\ v=1,\ldots,f. \tag{18}$$

In the problem $MinAct$, there are $k^*(f+n)$ variables and $1+n+k^*(1+2f+|E|+|A|)$ constraints without the 0-1 constraints.

Constraint (11) incorporates the lower bound LB_2. Constraints (12), though redundant because z_{vj} will be set to zero in an optimal solution if $x_{ij}=0$ for all $i\in N_v$, are included to be used by a solver in the solution process. Constraints (13) address the station capacities. Constraints (14) guarantee that any operation of a certain type can be assigned to a station only if this station is open for this type. Constraints (15) - (17) have the same meaning as in the problem $Min\#$.

3.3 Heuristics

For the problem $P(prec, excl, s_i \mid k, cost)$, we also suggest a randomized greedy heuristic, denoted as $GreedyCost$, to provide an upper bound. Let c_i denote the total activation cost of the operation $i\in N$: $c_i = \sum_{v\in T_i} a_v$. The only difference between the heuristics $GreedyCost$ and $GreedyNumber1$ is that in $GreedyCost$, instead of solving the $Subset\ Sum$ problem, the following $Knapsack$ problem is solved:

$$\max_{X\in N^+} \sum_{i\in X} c_i,\ \text{subject to} \sum_{i\in X} s_i \leq r.$$

An optimal solution of this problem assigns operations of maximum total cost to the same station.

Again, we use the standard dynamic programming algorithm to solve the $Knapsack$ problem. It requires $O(|N^+|r)$ time. Heuristic $GreedyCost$ can be independently run several times to increase the probability to obtain a feasible solution with a low cost.

Note that heuristic $GreedyCost$ can also be used to solve the problem $P(prec, excl, s_i \mid k)$ approximately, and heuristics $GreedyNumber1$ and $GreedyNumber2$ can be used to solve the problem $P(prec, excl, s_i \mid k, cost)$ approximately.

4 Computer experiments

We used ILOG CPLEX version 12.4 to solve the problems $Min\#$ and $MinAct$. The experiments were run on PC with Intel(R) Core(TM) Quad 2.83 Mhz processor and 8 Gb of RAM under MS Windows 7 Professional (64 bit). The data generation algorithm for CPLEX and the heuristics were coded in C++. In the experiments, the time limit for solving each instance of $Min\#$ or $MinAct$ was set to one hour.

Three families of instances were randomly generated. A family is associated with the number of operations $n \in \{40, 60, 80\}$. For all instances, there are three part types, the activation costs are $a_1 = 3$, $a_2 = 2$ and $a_3 = 1$ and the station capacity is equal to $r = 10$. Three part types are chosen because two or three modifications of the same product are typical in the application we are concerned with. Also in this application, 10 is one of the possible capacities of a magazine.

Each operation $i \in N$ is randomly assigned its size $s_i \in \{1, 2, 3\}$ and a set of types $T_i \in \Big\{\{1,2,3\},\{1,2\},\{1,3\},\{2,3\},\{1\},\{2\},\{3\}\Big\}$ with uniform probability distribution. The exclusion sets are defined as follows. For each pair (a, b) of operations such that they have at least one common type, it is decided with probability $1/3$ that a excludes b. Thus, all the exclusion sets are of cardinality two.

Each family is subdivided into four *series* n_S, n_M, n_L and n_X with small, medium, large and very large density of precedence constraints, respectively. For each series n_J, $J \in \{S, M, L, X\}$, and each pair (a, b) of operations such that $a < b$ and a and b have at least one common type, it is decided with probability p_J that a precedes b. The corresponding probabilities are $p_S = 0.2$, $p_M = 0.4$, $p_L = 0.6$ and $p_X = 0.8$. Since each arc in the obtained precedence graph always goes from a vertex with smaller index to a vertex with larger index, this graph is acyclic.

Each series consists of 20 instances. Their average numerical characteristics are given in Table 2. There, $\%Prec$ is the density of the precedence constraints defined as $\frac{|A|}{\max \#\text{ arcs}} \times$

$100\% = \frac{2|A|}{n^2-n} \times 100\%$.

Table 2: Average numerical characteristics

Series n_J	40_S	40_M	40_L	40_X	60_S	60_M	60_L	60_X	80_S	80_M	80_L	80_X
$\%Prec$	9.35%	19.25%	29.74%	40.25%	9.88%	19.74%	29.74%	41.24%	10.09%	19.93%	30.09%	39.43%
$\sum_{i=1}^{n} s_i$	78	79	79	80	117	120	120	120	160	160	158	159
$\|E\|$	189	186	202	199	444	451	443	458	797	790	791	786

The idea of these experiments is in line with similar studies on the transfer line balancing problems, see, e.g., Belmokhtar *et al.* [5], Borisovsky *et al.* [7] and Dolgui *et al.* [16]. There are no benchmark instances, however, because problems $Min\#$ and $MinAct$ have never been studied before.

The following procedure is used to solve each problem instance.

Solution procedure for the problem $P(prec, excl, s_i \mid k, cost)$:

Step 1 Apply heuristics $GreedyNumber1$, $GreedyNumber2$ and $GreedyCost$ 1000 number of times each. Determine the minimum number of stations, k^0, delivered by these heuristics.

Step 2 Apply CPLEX to solve the problem $Min\#$ with upper bound k^0 on the number of stations. Determine the minimum number of stations k^*.

Step 3 Apply CPLEX to solve the problem $MinAct$ with the number of stations k^*.

Step 4 Select the best solution for the problem $P(prec, excl, s_i \mid k^*, cost)$ among the solutions obtained in Steps 1, 2 and 3. ■

The possible outputs of each CPLEX run for an instance I are: 1) an optimal solution with value *Opt(I)* is found, 2) an approximate solution with value *Appr(I)* with a known relative error *%Gap(I)* and a lower bound *LB(I)* are obtained, 3) no feasible solution is found when the time limit is achieved. The relative error $\%Gap(I)$ is defined as $\frac{Appr(I)-LB(I)}{LB(I)} \times 100\%$. In our experiments case 3) never happened.

Table 3 shows the number of instances of problems $Min\#$ and $MinAct$ solved by CPLEX to optimality (row "*Opt*") and approximately (row "*Appr*"), for each series. Furthermore, the average (row "*Avg.Gap*") and the worst (row "*Worst Gap*") relative errors are listed for $Min\#$ problem. The average relative error *Avg.Gap* is defined as $\frac{1}{|n_J|} \sum_{I \in n_J} \%Gap(I)$ and the worst relative error *Worst Gap* as $\max_{I \in n_J} \{\%Gap(I)\}$ for each series n_J, $J \in \{S, M, L, X\}$. For the problem $Min\#$, 237 instances out of 240 were solved optimally and the remaining 3 instances were

solved approximately with an acceptable relative error. As for the problem $MinAct$, all instances were solved to optimality for all series. Table 3 also provides average (row "$Avg.Time$") and maximal (row "$Max.Time$") computational times of CPLEX application to problems $Min\#$ and $MinAct$ for all series.

Table 3: CPEX: solution quality and run time

Series n_J	40_S	40_M	40_L	40_X	60_S	60_M	60_L	60_X	80_S	80_M	80_L	80_X
	$Min\#$											
Opt	20	20	20	20	20	20	20	20	18	19	20	20
Appr	0	0	0	0	0	0	0	0	2	1	0	0
Avg.Gap	-	-	-	-	-	-	-	-	11.03	16.67	-	-
Worst Gap	-	-	-	-	-	-	-	-	13.1	16.67	-	-
Avg.Time (sec)	5.99	4.91	3.06	2.08	151.80	94.58	27.20	26.61	1416.60	1144.04	718.22	249.84
Max.Time (sec)	18.11	13.00	12.18	7.02	312.19	280.97	68.95	173.29	3600	3600	3137.18	1312.09
	$MinAct$											
Opt	20	20	20	20	20	20	20	20	20	20	20	20
Appr	0	0	0	0	0	0	0	0	0	0	0	0
Avg.Time (sec)	0.84	0.30	0.37	0.82	5.08	1.41	1.96	5.83	145.56	65.45	9.43	38.86
Max.Time (sec)	2.31	0.86	0.64	2.11	17.96	3.26	3.9	28.77	1037.45	1182.77	18.63	309.05

Performance of the heuristics *GreedyNumber1*, *GreedyNumber2* and *GreedyCost* is presented in Table 4. More precisely, the row *GreedyNumber AvgGap (GreedyNumber Worst.Gap)* provides the average gap (worst gap) of the best results founded on each instance with heuristics *GreedyNumber1*, *GreedyNumber2*. The section $Min\#$ of Table 4 contains the best results of the two heuristics, *GreedyNumber1* and *GreedyNumber2*, with respect to the number of stations. The section $MinAct$ of Table 4 shows the results of the heuristics *GreedyCost* with respect to the total activation cost. Two conclusions can be drawn from these results. First, for almost all instances where optimal solutions were found by CPLEX, values of heuristic solutions range between 10% and 40% of the optimum. Second, there is a high correlation between density of precedence constraints and heuristic solution quality. The higher the density, the better is the solution. It is logical since the higher density means a fewer number of alternate assignments to evaluate.

Table 4: Heuristics *GreedyCost* and *GreedyNumber*: solution quality and run time

Series n_J	40_S	40_M	40_L	40_X	60_S	60_M	60_L	60_X	80_S	80_M	80_L	80_X
	$Min\#$											
GreedyNumber Avg.Gap (%)	42.63	21.42	20.98	11.09	32.99	23.26	19.33	11.32	31.79	22.47	16.29	09.74
GreedyNumber Worst Gap (%)	66.67	38.53	37.57	25.52	46.15	39.83	32.78	20.90	50.12	39.35	31.67	24.16
Avg.Time (sec)	95.8	95.1	94.9	94.5	142.7	142.2	141.7	141.1	189.9	189.5	188.2	187.6
	$MinAct$											
GreedyCost Avg. Gap (%)	32.73	26.13	17.86	9.64	26.49	21.78	16.78	11.32	29.74	24.48	22.28	9.31
GreedyCost Worst Gap (%)	54.60	41.18	34.05	17.78	42.96	35.86	29.69	23.14	53.87	41.33	32.35	23.60
Avg.Time (sec)	125.6	125.2	124.7	124.3	202.8	202.3	201.9	201.4	342.1	341.6	341.1	340.8

General observations derived from the computer experiments are the following. First of all, CPLEX solves large scale instances of the problem $P(prec, excl, s_i \mid k, cost)$ in a short time. On average, problems are solved within 25 minutes. Secondly, problem $Min\#$ needs more computation effort than *MinAct*. It may be related to the high values of k^0 provided by the heuristics *GreedyNumber*. The third observation concerns the density of precedence constraints. More computation time is usually needed to solve instances with smaller density of precedence constraints.

We deduce that CPLEX can be used to solve instances of practical dimensions in acceptable time. Heuristics *GreedyNumber1* and *GreedyNumber2* can be used to obtain an initial number of stations k^0. All the heuristics can be used to obtain an approximate solution if the solution time is important. This is the case when there is a highly qualified and well paid expert whose job is to evaluate variants of the optimal line design for different types of equipment. Faster solutions save working time for the expert.

Note that earlier methods developed for similar problems observed in the line manufacturing industry were able to solve instances with up to 130 unit-sized operations, see, for example, Borisovsky et al. [7]. Since the average size of operations in the experiments of the current paper is two, the largest instances solved in this paper include about 160 unit size operations.

5 Dichotomic procedure

In this section, we show how to take advantage of the difference in computational times observed between both problems. As shown in the previous section, when CPLEX could not find the optimal solution for problem $Min\#$, it provided a lower bound on the minimum number of stations and an approximate solution, which delivers an upper bound on this value. Therefore, an optimal solution for the problem $P(prec, excl, s_i \mid k, cost)$ can be found by solving the problem $MinAct$ for each number of stations between these two bounds. In order to solve problem $P(prec, excl, s_i \mid k, cost)$ more efficiently in this case, we suggest the following dichotomic procedure.

Dichotomic solution procedure for the problem $P(prec, excl, s_i \mid k, cost)$ with a lower bound LB and an upper bound UB on the minimum number of stations:

Step 1 Apply CPLEX to solve the problem $MinAct$ with the number of stations $\lceil \frac{LB+UB}{2} \rceil$.

Step 2 Consider the following cases.

(a) A feasible solution (either optimal or with a known gap) is found. If $LB \neq UB$, then

reset $UB := \lceil \frac{LB+UB}{2} \rceil - 1$ and go to Step 1.

(b) CPLEX proves that there is no feasible solution. If $LB \neq UB$, then reset $LB := \lceil \frac{LB+UB}{2} \rceil + 1$ and go to Step 1.

(c) No feasible solution is found when the time limit is achieved. We have now two problems to solve:

c.1 If $LB \neq \lceil \frac{LB+UB}{2} \rceil$, then apply a new dichotomic procedure with a lower bound LB and an upper bound $\lceil \frac{LB+UB}{2} \rceil - 1$.

c.2 If $\lceil \frac{LB+UB}{2} \rceil \neq UB$, then apply a new dichotomic procedure with a lower bound $\lceil \frac{LB+UB}{2} \rceil + 1$ and an upper bound UB.

Step 3 Among the solutions obtained in Step 1 and 2, select the best solution for the problem $P(prec, excl, s_i \mid k, cost)$, which has the minimum cost among the solutions with the minimum number of stations. ■

The possible outputs of this procedure for an instance I of the problem $P(prec, excl, s_i \mid k, cost)$ are: 1) an optimal solution is found, 2) an approximate solution with a known relative error on the number of stations, or on the total cost, or on both criteria is obtained, 3) no new feasible solution is found when the time limit is achieved. In the latter case, however, the relative error can possibly be reduced if CPLEX proves that no feasible solution exists for some of the numbers of stations considered.

Let us now illustrate this procedure on the three instances that remained unsolved in the previous section (see Tables 5, 6, 7).

Table 5: Illustration of dichotomic procedure on the first instance unsolved for the series 80_S

Use of the dichotomic procedure with $LB = 18$ and $UB = 19$:
Apply CPLEX to solve the problem $MinAct$ with the number of stations 19
Output: the optimal solution is found in 318 sec.
Use of the dichotomic procedure with $LB = 18$ and $UB = 18$:
Apply CPLEX to solve the problem $MinAct$ with the number of stations 18
Output: the optimal solution is found in 389 sec.
Procedure Output: the optimal solution of the instance is found in 707 sec.

As shown in these three examples, this dichotomic procedure can in some cases provide better solutions. However, the resolution of the problem $MinAct$ appears much more difficult when CPLEX cannot find a feasible solution. Overall, the procedures described in this paper have

Table 6: Illustration of dichotomic procedure on the second instance unsolved for the series 80_S

Use of the dichotomic procedure with $LB = 18$ and $UB = 20$:
Apply CPLEX to solve the problem $MinAct$ with the number of stations 19
Output: the optimal solution is found in 679 sec.
Use of the dichotomic procedure with $LB = 18$ and $UB = 18$:
Apply CPLEX to solve the problem $MinAct$ with the number of stations 18
Output: no feasible solution found within the time limit.
Procedure Output:
An approximate solution with 19 stations instead of 20 is found in 4,279 sec.
A gap of 1 station (i.e., 5.56%) instead of 2
For a line with 19 stations, the optimal total cost has been found

Table 7: Illustration of dichotomic procedure on the instance unsolved for the series 80_M

Use of the dichotomic procedure with $LB = 26$ and $UB = 30$:
Apply CPLEX to solve the problem $MinAct$ with the number of stations 28
Output: the optimal solution is found in 1,233 sec.
Use of the dichotomic procedure with $LB = 26$ and $UB = 27$:
Apply CPLEX to solve the problem $MinAct$ with the number of stations 27
Output: no feasible solution found within the time limit.
Use of the dichotomic procedure with $LB = 26$ and $UB = 26$:
Apply CPLEX to solve the problem $MinAct$ with the number of stations 26
Output: no feasible solution found within the time limit.
Procedure Output:
An approximate solution with 28 stations instead of 30 is found in 8,433 sec.
A gap of 2 stations (i.e., 7.69%) instead of 4
For a line with 28 stations, the optimal total cost has been found

permitted to solve optimally 238 instances out of 240, and the 2 remaining instances were solved approximately with a gap of 1 and 2 workstations respectively (and the optimal total cost for the corresponding number of stations).

6 Conclusions

In this paper, we studied the line balancing problem $P(prec, excl, s_i \mid k, cost)$ in which a line has to be configured to produce parts of different types. Each part of a specific type moves between stations in the same direction and a station is activated if at least one operation is assigned to it. Operations on the same part assigned to the same station are launched simultaneously and executed in parallel. The primary objective is to minimize the number of stations and the secondary objective is to minimize the total activation cost. Several results on computational complexity have been presented for this problem and are summarized in Table 8.

Table 8: Computational complexity

Problem	Complexity
$P(prec, s_i = 1 \mid k)$, arbitrary r	strongly NP-hard
$P(prec, s_i = 1 \mid k)$, constant r	open
$P(in-tree, s_i = 1 \mid k)$	$O(n)$
$P(out-tree, s_i = 1 \mid k)$	$O(n)$
$P(prec, s_i = 1 \mid k)$, $r = 2$	$O(n^2)$
$P(s_i \mid k)$, arbitrary r	strongly NP-hard
$P(s_i \mid k)$, constant r	$O(n^{const})$
$P(excl, s_i = 1 \mid k)$	strongly NP-hard
$P(s_i = 1 \mid k, cost)$	strongly NP-hard

In order to solve the general case of the problem $P(prec, excl, s_i \mid k, cost)$, we suggested a four-stage solution approach, which combines heuristic algorithms with integer linear programming. Computer experiments show that this approach is able to solve instances of practical sizes. A dichotomic procedure has also been presented and has permitted to obtain better solutions in the few cases where the four-stage solution approach was not able to solve the problem to optimality.

It is also interesting to note that the proposed dichotomic procedure can be used for solving the problem $P(prec, excl, s_i \mid k, cost)$ without solving the problem $Min\#$. For example, LB_1 or the optimal value of the continuous relaxation of $Min\#$ can be used as a lower bound, and the best solution provided by the heuristics can be used as an upper bound. In this case, the dichotomic procedure can possibly require the resolution of a few more $MinAct$ problems, but these additional resolutions can consume less time. Considering the results obtained for the examples presented in Section 5, it appears that the quality of the lower bound is more critical for the computational time of the dichotomic procedure than the quality of the upper bound. As a consequence, besides a search for more efficient heuristics, improvement of the lower bound for the problem $Min\#$ seems to be a more promising way to solve problems of larger dimensions.

References

[1] Akpinar, S. and Mirac Bayhan, G. (2011) A hybrid genetic algorithm for mixed model assembly line balancing problem with parallel workstations and zoning constraints. *Engineering Applications of Artificial Intelligence*, **24**, 449-457.

[2] Andres, C., Miralles, C., and Pastor, R. (2008) Balancing and scheduling tasks in assembly

lines with sequence-dependent setup times. *European Journal of Operational Research*, **187**, 1212-1223.

[3] Battaia, O., and Dolgui, A. (2012) Reduction approaches for a generalized line balancing problem. *Computers and Operations Research*, **39**, 2337–2345.

[4] Battaia, O., and Dolgui, A. (2013) A taxonomy of line balancing problems and their solution approaches. *International Journal of Production Economics*, **142(2)**, 259–277.

[5] Belmokhtar, S., Dolgui, A., Guschinsky, N., and Levin, G. (2006) Integer programming models for logical layout design of modular machining lines. *Computers & Industrial Engineering*, **51(3)**, 502-518.

[6] Borgia, S., Matta, A., and Tolio, T. (2013) STEP-NC compliant approach for setup planning problem on multiple fixture pallets. *Journal of Manufacturing Systems*, in Press, DOI: 10.1016/j.jmsy.2013.09.002.

[7] Borisovsky, P., Dolgui, A., and Kovalev, S. (2012) Algorithms and implementation of a set partitioning approach for modular machining line design. *Computers and Operations Research*, **39**, 31473155.

[8] Boysen, N., and Fliedner, M. (2008) A versatile algorithm for assembly line balancing. *European Journal of Operational Research*, **184**, 39-56.

[9] Boysen, N., Fliedner, M. and Scholl, A. (2009) Sequencing mixed-model assembly lines: Survey, classification and model critique. *European Journal of Operational Research*, **192**, 394-373.

[10] Boysen, N., Scholl, A., and Wopperer, N. (2012) Resequencing of mixed-model assembly lines: Survey and research agenda. *European Journal of Operational Research*, **216**, 594-604.

[11] Bukchin, Y., and Rabinowitch, I. (2006) A branch-and-bound based solution approach for the mixed-model assembly line-balancing problem for minimizing stations and task duplication costs. *European Journal of Operational Research*, **174(1)**, 492-508.

[12] Brucker, P. and Knust, S. http://www.mathematik.uni-osnabrueck.de/research/OR/class.

[13] Caprara, A. and Pferschy, U. (2005) Modified subset sum heuristics for bin packing. *Information Processing Letters*, **96(1)**, 18-23.

[14] Davida, G.I., and Linton, D.J. (1976) A new algorithm for the scheduling of tree structured tasks in *Proc. Conf. Inform. Sci. and Syst.*, Baltimore MD, pp. 543-548.

[15] Emse, S., Boysen, N., and Scholl, A. (2010) Balancing mixed-model assembly lines: A computational evaluation of objectives to smoothen workload. *International Journal of Production Research*, **48(11)**, 3173-3191.

[16] Dolgui, A., Kovalev, S., Kovalyov, M.Y., Nossack, J., and Pesch, E. (2014) Minimizing setup costs in a transfer line design problem with sequential operation processing. *International Journal of Production Economics*, **151**, 186-194.

[17] Dolgui, A., Guschinsky, N., Levin, G., and Proth, J.M. (2008) Optimisation of multi-position machines and transfer lines. *European Journal of Operational Research*, **185(3)**, 1375-1389.

[18] Dolgui, A., Finel, B., Guschinsky, N., Levin, G., and Vernadat, F. (2006) MIP approach to balancing transfer lines with blocks of parallel operations. *IIE Transactions*, **38(10)**, 869-882.

[19] Dolgui, A., and Proth, J.-M. (2010) *Supply chain engineering: useful methods and techniques*, Springer.

[20] Erel, E., and Sarin, S.C. (2009) A survey of the assembly line balancing procedures. *Production Planning and Control*, **120** 276-286.

[21] Falkenauer, E. (2005) Line balancing in the real world in *Proceedings of the International Conference on Product Lifecycle Management PLM'05*, pp. 360-370.

[22] Fernandez de la Vega, W., and Lueker, G. (1981) Bin Packing can be solved within $(1+\varepsilon)$ in linear time. *Combinatorica*, **1(4)**, 349-355.

[23] Gabow, H.N. (1982) An almost linear algorithm for two processor scheduling. *Journal of the ACM*, **29(3)**, 776-780.

[24] Gamberini, R., Grassi, A., and Rimini, B. (2006) A new multi-objective heuristic algorithm for solving the stochastic assembly line re-balancing problem. *International Journal of Production Economics*, **102** 226-243.

[25] Garey, M.R., and Johnson, D.S. (1979) *Computers and intractability. A guide to the theory of NP-completeness*, W.H. Freeman and Company, San Francisco.

[26] Haouari, M., and Serairi, M. (2009) Heuristics for the variable sized bin-packing problem. *Computers & Operations Research*, **36 (10)**, 2877-2884.

[27] Herrmann, J.W. (2012) Finding optimally balanced words for production planning and maintenance scheduling. *IIE Transactions*, **44 (3)**, 215-229.

[28] Hu, T.C. (1961) Parallel sequencing and assembly line problems. *Operations Research*, **9**, 841-848.

[29] Kabir, Md.A., and Tabucanon, M.T. (1995) Batch-model assembly line balancing: A multiattribute decision making approach. *International Journal of Production Economics*, **41**, 193-201.

[30] Kovalev, S., Delorme, X., and Dolgui, A. (2012) Line configuration to minimize setup costs. *Mathematical and Computer Modelling*, **55**, 2087-2095.

[31] Matanachai, S., and Yano, C.A. (2001) Balancing mixed-model assembly lines to reduce work overload. *IIE Transactions*, **33(1)**, 29-42.

[32] McMullen, P.R., and Tarasewich, P. (2005) A beam search heuristic method for mixed-model scheduling with setups. *International Journal of Production Economics*, **96**, 273-283.

[33] McMullen, P.R., and Tarasewich, P. (2006) Multi-objective assembly line balancing via a modified ant colony optimization technique. *International Journal of Production Research*, **44**, 27-42.

[34] Potts, C.N., and Kovalyov, M.Y. (2000) Scheduling with batching: a review. *European Journal of Operational Research*, **120**, 228-249.

[35] Rekiek, B., Dolgui, A., Delchambre, A., and Bratcu, A. (2002) State of art of assembly lines design optimization. *Annual Reviews in Control*, **26**, 163-174.

[36] Sawik, T. (2005) Integer programming models for the design and balancing of flexible assembly systems. *Mathematical and Computer Modelling*, **21**, 1-12.

[37] Suwannarongsri, S., and Puangdownreong, D. (2008) Multi-objective assembly line balancing via adaptive tabu search method with partial random permutation technique. *IEEE International Conference on Industrial Engineering and Engineering Management*, 312-316.

[38] Ullman, J.D. (1975) NP-complete scheduling problems. *Journal of Computer and System Sciences*, **10**, 384-393.

[39] Vilarinho, P.M., and Simaria, A.S. (2006) ANTBAL: An ant colony optimization algorithm for balancing mixed-model assembly lines with parallel workstations. *International Journal of Production Research*, **44**, 291-303.

[40] Yan, H-S., Xia, Q-F., Zhu, M-R., and Liu, X-L. (2003) Integrated Production Planning and Scheduling on Automobile Assembly Lines. *IIE Transactions*, **35(8)**, 711-7253

[41] Yuan, B., Zhang, C., Shao, X., and Jiang, Z. (2015) An effective hybrid honey bee mating optimization algorithm for balancing mixed-model two-sided assembly lines. *Computers and Operations Research*, **53**, 32-41