Branch and Bound Algorithm
for a Transfer Line Design Problem:
stations with sequentially activated multi-spindle heads

A. Dolgui\textsuperscript{1,∗}, and I. Ihnatsenka\textsuperscript{1,2,†}

\textsuperscript{1} Ecole des Mines de Saint-Etienne, France

\textsuperscript{2} Grodno State University, Belarus

Abstract

In this paper a new transfer line balancing problem is considered. The main difference from the simple assembly line balancing problem is that the operations are grouped into blocks. All the operations of the same block are carried out simultaneously by one piece of equipment (multi-spindle head). Nevertheless, the blocks assigned to the same workstation are executed in series. The line cost consists of the sum of block and workstation costs. At the considered line design stage, the set of all available blocks is given. The aim is to find the subsets of blocks to be assigned to each workstation such that each operation is executed once and the line cost is minimal while all the precedence, cycle time, and compatibility constraints are respected. A new lower bound based on solving a special set partitioning problem is presented and a branch-and-bound algorithm is developed. The experimental results prove the quality of the lower bound and applicability of the suggested approach for medium sized industrial cases.

Key words: line balancing, lower bound, branch-and-bound, set partitioning.

\textsuperscript{∗}Corresponding author: e-mail: dolgui@emse.fr, tel.: +33 4 77 42 01 66, address: 158, cours Fauriel, 42023, Cedex 2, France

\textsuperscript{†}Author is supported by a NATO grant
1. Introduction and motivation

Flow lines with workstations in series are frequently used in manufacturing systems. They increase the production rate and minimize the cost of produced items [18, 17]. This paper deals with a flow line design problem in machining/process environment. It concerns the automatic tandem lines (transfer lines or transfer machines) without buffers often used in automotive industry. The transfer lines are designed for mass production of a single type of part (several million parts for a year). Ordinarily, they have large investment costs (between 5 million and 50 million euros).

In the transfer lines, the operations are grouped into blocks. All operations of each block are executed by a multi-spindle head (a piece of equipment with several spindles, where each spindle executes one operation). An illustration of such a transfer line is presented in Figure 1. The spindle heads of the same station are activated in series in a given order. In this figure, the numbers on spindle-heads also give the order of their activation. At station 2, first, spindle head 1 executes its operations, then spindle head 2 is activated, and finally spindle head 3. The cycle time is finished and the part is moved to station 3 by the conveyor and the next cycle begins.

[Insert Figure 1 about here]

To simplify this figure, we presented the spindle heads with only one, two or three tools (which execute one, two or three operations, respectively). In reality, the spindle heads can contain fifteen or more spindles (tools). In Figure 2, we show a station with two sequentially activated spindle heads, where each spindle head is composed of more than ten spindles. At this station, first the horizontal spindle head is activated to execute the 13 drilling operations simultaneously, then by rotating the spindle head support, the vertical spindle head is positioned in the horizontal axis, and executes ten other operations again simultaneously.

[Insert Figure 2 about here]

The integration of operations into blocks essentially increases the productivity of the line [17]. An illustration of this claim is given on Figure 3. It is necessary to drill two bores (operation \(A\) and operation \(B\)). The duration is 40 for operation \(A\) and is 60 for operation \(B\). If the operations are executed in series then their total duration is the sum of operation times 40 + 60 = 100 (see Figure 3a). If the operations are executed simultaneously then their total time is a maximum of operation durations \(\max(40, 60) = 60\) (see Figure 3b). Therefore, the profit is 40.
In brief, all spindles of a spindle head are activated *simultaneously* executing all the operations in parallel. Therefore, the multi-spindle heads are crucial elements of transfer lines, they:

i) provide a high line productivity;
ii) are complex mechanically;
iii) are very expensive.

Therefore, it is necessary to seriously examine all possible structures of spindles heads while analyzing their costs and productivity at the design stage. Due to the fact that the cost of spindle heads is an important part in the total cost of the line as well as the line exploitation period is relatively long (minimum 7 years, often 30 years with some modification every 7 years), this analysis is crucial economically.

Basically, the design of these lines involves the following five interconnected steps:

i) the selection of operations to machine each element of a part and a calculation of basic operation parameters;
ii) the selection of the type of machining system (with sequential, parallel or mixed activation of spindle heads);
iii) the logical synthesis of the manufacturing process which consists in grouping the operations into blocks, the analysis of feasibility of the corresponding spindle heads and their costs;
iv) the logical layout, i.e. grouping the blocks into stations taking into account the productivity and technological constraints;
v) the design or procurement of the equipment (tools, spindle heads, etc.) depending on the logical layout contained.

If at a given step there is no acceptable decision, the designer comes back to the previous steps, i.e., these steps may be performed several times, iteratively.

The decision on the type of transfer line is made at the second step depending on several economical and technical factors. The selected activation mode for spindle heads significantly influences the features of a transfer line. In this paper, we consider the line with the sequential activation of spindle heads at each station. For these lines the maximal available work time per workstation is limited by a given *line cycle time*.

The paper deals with the fourth step, i.e. the logical layout design. It is assumed that all the results of the first three steps and the constraints of the fifth step,
which must be considered at the fourth step, are known. The design problem is to
determine sets of blocks to be assigned to workstations so that the line cost is minimal
and all the constraints are respected. We refer to this problem as the Transfer Line
Balancing Problem (TLBP) by analogy with the Simple Assembly Line Balancing
Problem (SALBP).

The high investment cost of a transfer line demands an optimal solution. However,
optimization of such a complex system is a very difficult combinatorial problem. The
SALBP is proven to be an NP-hard problem [25]. Therefore, the problem under
consideration is NP-hard as well because it is a generalization of SALBP.

This paper is organized as follows. Section 2 presents a short state of the art on
Line Balancing Problems. Section 3 gives the statement of the new line balancing
problem considered and its mathematical model. Section 4 presents an original lower
bound based on a special set partitioning problem. Section 5 is dedicated to the de-
velopment of an efficient branch-and-bound algorithm. Finally, experimental results
are discussed in Section 6.

2. Related works

The typical flow line balancing (LB) problem consists of assigning a given set of
operations to workstations taking into account precedence relations between opera-
tions while minimizing the total number of workstations for the given line cycle time
(the solution with the minimum number of stations has a minimum idle time). Such
a problem is known in literature as the time-oriented simple assembly LB problem
(SALBP). Its detailed analysis is given in [25].

The SALBP has been intensively studied and a great number of approaches like
dynamical programming, branch and bound algorithms, heuristics and metaheuristics
have been proposed. Comprehensive surveys on SALBP are given in [4, 15, 23, 24,
25, 26].

Another category of problems is the cost-oriented simple assembly line balancing
problem (COALBP)[2, 3]. For SALBP, it is supposed that all stations (and workers)
are identical. Thus, the goal is to minimize their number. For COALBP, it is assumed
that the wages of workers depend on their skills. The different operations require
different worker qualifications. Therefore, the minimization of workstations cannot
be the primary objective. In this case, to minimize the cost per produced unit, it is
necessary to consider the wages of workers. For the COALBP, the optimal solution is
to insure those workers that are the highest paid are better utilized. This approach
is interesting for manual assembly lines with large differences among the levels of worker qualifications.

There are other generalizations of SALBP, like the line design problem with equipment selection (SALDP). This problem consists of assigning simultaneously the operations and pieces of equipment to workstations [6, 7, 16]. This is by definition a problem with many more parameters and thus more complicated. Indeed, for SALBP the time of an operation does not depend on the workstation where this operation is executed. For SALDP, the operation times depend on the equipment selected. This problem is characteristic of robotic assembly lines with sequential execution of operations.

There are only a few papers which deal with line balancing problem in the machining/process environment [5, 9, 10, 11, 12, 13, 14]. In the papers [9, 10, 12, 13], the set of all available spindle heads is not known beforehand, i.e. blocks are decision variables. The goal is to minimize a weighted sum of workstation and spindle head numbers. This problem can be considered as an element of step 3 of the general procedure explained earlier (in introduction), i.e. the generation of all possible (and interesting) solutions for spindle heads.

The papers [5, 11, 14] have been devoted to logical layout design where set of blocks is given and blocks at each station are activated simultaneously. Several approaches were proposed: mixed integer programming (MIP), a constrained shortest path and heuristics.

This paper deals with an as yet unexplored TLBP for the case where blocks assigned to workstation are executed in series and the set of all the available blocks is given. For each block, its cost and execution time are given. The objective is to minimize the line investment cost composed of block and workstation cost. More precisely, it is necessary to choose a subset from the given set of available blocks and to find a partition of this subset to workstations so that the line investment cost is minimal. The obtained subset of blocks must cover all the operations to be executed and must respect all constraints.

This problem cannot be directly solved by the SALBP approaches and new methods based on a wide range of combinatorial optimization techniques are needed for the following reasons:

- operations are grouped into the blocks and cannot be separately assigned to workstations (in contrast to SALBP);
- several blocks can contain the same operation and in general it is not known
beforehand which blocks are better;

- several additional constraints described in the next section must be taken into account.

In the next section, we present this problem formally.

3. **Problem Statement**

This transfer line balancing problem consists of assigning blocks from the given set to workstations in such a way that:

i) each operation must be assigned once (to one workstation only);

ii) each workstation time is not greater than the given line cycle time;

iii) the precedence constraints which are defined for the set of all operations are not violated. Because the operations are grouped into the blocks, the following situation is possible: block $A$ having an operation which must precede an operation of block $B$, and, at the same time, block $B$ having an operation which must precede an operation of block $A$. These precedence constraints cannot be used to reduce the number of available blocks because there can exist a feasible solution contained either block $A$ or block $B$.

iv) some groups of operations must be performed at the same workstation (i.e. they must be in different blocks assigned to the same workstation or in the same assigned block). This type of constraint is called *operation inclusion constraint* [25]. They are close but not similar to well-known zoning constraints in SALBP. Zoning constraints appear if an operation must be assigned to a given subset of workstations (relation ”operation-station”), inclusion constraints concern only the relations between operations (operations which cannot be assigned to the same station no matter what station);

v) some groups of blocks cannot be carried out at the same workstation. This type of constraint is called *block exclusion constraint*. There are publications for SALBP where the exclusion constraints exist. This means that some operations cannot be executed at the same workstation. These relations can be easily transferred to TLBP. Two blocks cannot be executed at the same workstation if they contain incompatible operations. Thus, operation exclusion constraints define block exclusion constraints. However, even if two blocks contain only compatible operations, these blocks can be incompatible for technological reasons. Therefore, in a real life situation, the block exclusion constraints are more general than the operation exclusion constraints;
vi) the number of blocks assigned to the same workstation does not exceed the
  given value $n_0$ (limited total power, total feed force, space, ...);

vii) the cost of the transfer line, evaluated as the sum of assigned block and occupied
  workstation costs, is to be minimized.

The following notation is used:

$N = \{1, 2, \ldots, n\}$ is the given set of operations to be assigned;

$T_0$ is the given transfer line cycle time;

$n_0$ is the maximal number of blocks per workstation;

$q_0$ is the number of all possible blocks;

$B = \{b_1, b_2, \ldots, b_q\}$ is the given set of possible blocks;

$N(b) \subseteq N$ is the set of operations which are included in block $b$;

$t(b)$ is the processing time of block $b \in B$;

$c(b)$ is the cost of block $b \in B$;

$C_0$ is the cost of one workstation;

$Pr(b)$ is the set of operations which must directly precede each operation from $N(b)$;

$\sim$ is the inclusion relation between two operations: $i \sim j$, $i, j \in N$ iff operation $i$
  and operation $j$ either must be assigned to two different blocks of the same
  workstation or must be included in the same assigned block. Clearly, this
  relation is transitive. In another way, operation inclusion constraints can be
  defined by graph $G^o = (N, E^{op})$, $E^{op} = \{(i, j) \mid i, j \in N, i \sim j\}$;

$\sim$ is the exclusion relation between two blocks: $b' \sim b''$, $b', b'' \in B$ iff block $b'$ and $b''$
  can not be assigned to the same workstation;

$I^o(i)$ is a set of operations $j$ such that $i, j \in N, i \sim j$, for all $j \in I^o(i)$;

$G^r = (N, D^r)$ is an acyclic digraph representing the precedence constraints for oper-
  ations. Arc $(i, j) \in D^r$ iff operation $i$ precedes operation $j$;

$G^{BE} = (B, E^{BE})$ is a graph representing the block exclusion constraints:

$$E^{BE} = \{(b', b'') \mid \text{iff } b' \sim b'', b', b'' \in B\};$$

$m$ is the number of workstations in the obtained solution;

$q_i$, $i = 1, 2, \ldots, m$ is the number of assigned blocks to the workstation with index $i$;

$LW_S$ is a lower bound for the given sequence $S$. The sequence $S$ is a solution of the
  TLBP.
The TLBP with sequentially executed blocks can be formulated as follows: to find a sequence of blocks 
\[ S = \{s_{11}, \ldots, s_{1q}, \ldots, s_{m1}, \ldots, s_{mqm}\}, s_{ku} \in B, k = 1, 2, \ldots, m, \]
u = 1, 2, \ldots, qm \] such that the following constraints are respected:

\[ \bigcup_{k=1}^{m} \bigcup_{u=1}^{q_k} N(s_{ku}) = N, \] (1)

\[ N(s_{ku}) \cap N(s_{rv}) = \emptyset, \quad ku \neq rv, \] (2)

for all \( s_{ku} \in S \), \( Pr(s_{ku}) \subseteq \left( \bigcup_{r=1}^{k-1} \bigcup_{v=1}^{q_r} N(s_{rv}) \right) \bigcup \left( \bigcup_{v=1}^{u} N(s_{kv}) \right) \). (3)

\[ \bigcup_{u=1}^{q_k} \bigcup_{i \in N(s_{ku})} P^i(i) \subseteq \bigcup_{u=1}^{q_k} N(s_{ku}), \quad k = 1, 2, \ldots, m, \] (4)

\[ (s_{ku}, s_{kv}) \notin E^{BE}, \quad k = 1, 2, \ldots, m; \quad u, v = 1, 2, \ldots, q_k, \] (5)

\[ \sum_{u=1}^{q_k} t(s_{ku}) \leq T_0, \quad k = 1, 2, \ldots, m, \] (6)

\[ q_k \leq n_0, \quad k = 1, 2, \ldots, m, \] (7)

and the objective function (transfer line cost)

\[ C(S) = C_0 m + \sum_{k=1}^{m} \sum_{u=1}^{q_k} c(s_{ku}) \] (8)

is to be minimized.

Constraints (1) and (2) provide that all the operations from \( N \) are assigned and that each operation is executed once. Expressions (3) define the precedence constraints between operations. Constraints (4) and (5) are operation inclusion and block exclusion constraints, respectively. Constraints (6) provide that workstation times do not exceed the given line cycle time. Conditions (7) mean that each workstation can contain at most \( n_0 \) blocks. An instance of this problem (1)-(8) is given in Figure 4.

In the following sections, we propose an original procedure to solve this hard combinatorial problem. This procedure is based on decomposition illustrated in Figure 5. There are three embedded procedures. The first procedure finds a lower bound on
the number of stations. The result of this procedure is used in the second, which is
a backtrack algorithm to solve optimally a special set partitioning problem. The
optimal solution obtained by the second procedure is used in the main procedure
(branch and bound algorithm) as a lower bound for the objective function of the
initial problem (1)-(8).

4. Lower Bound

The common approach to obtain a lower bound is to exclude some constraints from
the initial model. This defines a new optimization problem which must be simpler
than the initial. If new problem has a solution then its minimal cost is a lower bound
of the initial problem.

The proposed paper is in this framework. It is suggested to obtain a lower bound
by considering the constraints (2) (selected blocks are disjoint) and constrains (1)
(all the operations are assigned). These constraints appear in the well-known set
partitioning problem. It can be formulated by following. There is a family of subsets
of the given set. Each subset has a cost. It is necessary to find a set of disjoint subsets
such that their union equals the given set and the total cost is minimal. It was studied,
for example, in [19, 20, 21]. All these publications focused on tree-search algorithms
used directly or combined with linear programming. The set partitioning problem is
NP-complete. However, it can be efficiently solved in practice for instances with a
reasonable size.

This fact suggests to obtain a lower bound using the set partitioning problem.
However, even if the constraints (3)-(7) are discarded, the new optimization problem
differs from the standard set partitioning problem by the cost of occupied work-
stations $C_0m$ in the objective (8). This cost depends on the number of occupied
workstations $m$. In general, the computation of the number $m$ is a complex combi-
natorial problem. For example, even if the blocks of an optimal solution are known
then this problem is not easily than a bin-packing problem.

Another approach for obtaining lower bound is to estimate the cost of unassigned
blocks. However, in most TLBP $C_0$ equals 150% of the average cost of blocks. There-
fore, the cost of occupied workstations can be essential and its discarding causes that
the gap between the lower bound and the optimal objective could increase.

The lower bound developed in this paper provides a compromise between the
complexity and exactness. The lower bound is based on a special set partitioning

[Insert Figure 5 about here]
problem. The estimation of the number of occupied workstations is carried out by the cycle time and block exclusion as well as block number constraints. The rest of this section explains the suggested approach.

Let $S$ be a sequence which satisfies the constraints (2), (3), (5)-(7) and the constraint (4) for $k = 1, 2, \ldots, m-1$. The constraint (1) is not respected (otherwise, there is no necessity to find a lower bound). These conditions mean that all the workstations with number smaller than $m$ are completely determined. The last workstations is known partially and the rested workstations are unknown.

Let introduce the following notations:

- $N(S) = \bigcup_{s \in S} N(s)$ be a set of processed operations after executing all the blocks $s \in S$;
- $\hat{N}(S) = \bigcup_{U=1}^{q_m} N(s_m u)$ be a set of operations assigned to the last workstation of a sequence $S$;
- $\overline{N}(S) = N \setminus N(S)$ be a set of unassigned operations.

Let $X$ be a set of disjoint blocks which satisfies the following constraints:

- contains all the blocks assigned to the last workstation of a sequence $S$; 
- none of rested blocks contains assigned operations by a sequence $S$.

Formally, these constraints are equivalently written as follows:

$$X \subseteq B, \quad s_{mu} \in X, \ u = 1, 2, \ldots, q_m, \quad N(x) \subseteq \overline{N}(S) \cup \hat{N}(S), \ x \in X \quad (9)$$

The set $N(X) = \bigcup_{x \in X} N(x)$ is considered as a state of the set $X$.

Evidently, the cost of the blocks of a set $X$ equals $\sum_{s \in X} c(x)$. As explained earlier, it is necessary to estimate the number of workstations required to assign all the blocks from a set $X$. Let $\tilde{m}$ be a lower bound of the number of required workstations. In this paper, the number $\tilde{m}$ is estimated by considering the constraints (5)-(7).

Firstly, the block exclusion constraints (5) are considered. A set $X$ induces in the block exclusion graph $G^{BE}$ a subgraph $G^X = (X, E^X), E^X = \{(u, v) \mid u, v \in X, (u, v) \in E^{BE}\}$. Let $G^X$ be the complement of graph $G^X$ and $G^X_k = (V_k, E^X_k), \ k = 1, 2, \ldots, l_X$ are the interconnected components of $G^X$. Indeed, if two nodes from $X$ are in different
components, then the corresponding blocks cannot be assigned to the same workstation. Therefore, the constraint (5) reformulates the problem of \( \tilde{m} \) estimation: to obtain \( \tilde{m} \) it is sufficient to calculate the number of required workstations \( m_k \) for each component \( G_k^X \).

Now, consider the component \( G_k^X = (V_k, E_k^V) \), \( k = 1, 2, \ldots, l_X \). A set \( V_k \) induces in \( G^X \) a subgraph \( (V_k, E_k^V) \). If a subset of nodes \( \tilde{V}_k \subseteq V_k \) of this subgraph belongs in totality to the same clique (by report \( E_k^V \)), then the blocks which correspond to the nodes of \( \tilde{V}_k \) cannot be assigned to the same workstation.

Therefore, the minimal number of required workstations is greater or equals than \( \omega(G^X) \), where \( \omega(G) \) is the clique number of a graph \( G \). A lower bound of the clique number \( \tilde{\omega}(G_k^X) \) for a component \( G_k^X \) can be computed as follows [1]:

\[
\tilde{\omega}(G_k^X) = \sum_{v \in V_k} \frac{1}{|V_k| - \text{deg}(v)},
\]

where \( \text{deg}(v) \) is the degree of a node \( v \) in a subgraph \( G_k^X \). Due to the fact that \( m \) is integer, it is required at least \( \lceil \tilde{\omega}(G_k^X) \rceil \) workstations to assign the blocks of the set \( V_k \).

An illustration is given in Figure 6. In Figure 6a an original block exclusion graph is presented. Figure 6b gives the complement graph which contains two components (outlined by dashed frameboxes). Figure 6c shows the subgraphs \( G_k^X \) induced by these components. Their clique numbers are equal 1 and 4/3 respectively. Then, at least three workstations are necessary for this example.

[Insert Figure 6 about here]

Now, consider the cycle time (6) and block number (7) constraints. Evidently, they lead to the following estimates:

\[
m_k \geq \left\lceil \frac{\sum_{v \in V_k} t(v)}{T_0} \right\rceil, \quad m_k \geq \left\lceil \frac{|V_k|}{n_0} \right\rceil.
\]

Using the above inequalities and the estimate by the clique number, a lower bound of the number of workstations for a subgraph \( G_k^X \) can be obtained as follows:

\[
\tilde{m}_k = \max \left\{ \left\lceil \tilde{\omega}(G_k^X) \right\rceil, \left\lceil \frac{\sum_{v \in V_k} t(v)}{T_0} \right\rceil, \left\lceil \frac{|V_k|}{n_0} \right\rceil \right\}.
\]

(10)

Finally, the lower bound of the number of workstations for set \( X \) is computed as:

\[
\tilde{m} = \sum_{k=1}^{l_X} \tilde{m}_k.
\]

(11)
Therefore, to obtain lower bound for the objective (8) for the given sequence $S$, it is necessary to find a set $X$ which satisfies constraints (9) as well as

$$
\mathcal{N}(X) = \overline{\mathcal{N}}(S) \bigcup \hat{\mathcal{N}}(S),
$$

and minimizes the function:

$$
LW(X) = \sum_{s \in S} c(s) + \sum_{x \in X, x \notin S} c(x) + (m - 1 + \tilde{m})C_0,
$$

Denote $LW_S$ is the optimal value of function $LW(X)$ for the given sequence $S$.

The problem (9), (12), (13) is a special set partitioning problem. It differs from the standard set partitioning problem by the objective (13) (in case of standard problem the objective is a linear function). Firstly, the objective (13) is nonlinear and does not have an analytical expression (it is defined by an algorithm). Secondly, objective (13) depends not only on block costs but on the number of blocks and their execution times as well. These features make the procedures suggested in [19, 21] to be inappropriate in this case.

However, function (13) is isotonic (if sets $X' \subseteq X''$, then $LW(X') \leq LW(X'')$). Therefore, all the tree search approaches are suitable to solve the set partitioning problem (9), (12), (13). In this paper, an adaptation of the tree search algorithm proposed by R. Garfinkel and G. Nemhauser is presented.

The approach developed in [20] is a simple backtracking algorithm. Some of its details are described below:

- The special ordering of blocks is applied before the algorithm starts. Namely, blocks are partitioned into lists $L_i$, $i = 1, \ldots, n$. List $L_i$ contains all the blocks which have operation $i$ and do not have operations $1, 2, \ldots, i - 1$ (some lists can be empty).
- The cost of the initial complete (containing all the operations to be assigned) solution is supposed to be infinite. The algorithm starts with an empty partial solution. It corresponds to the root node of the search tree.
- The minimal number $i^*$ of unassigned operations is computed for the current partial solution. The list $L_i^*$ is fixed as the current one. Each block from $L_i^*$ contains the operation $i^*$ and no blocks from the lists with greater number than $i^*$ contains the operation $i^*$ (by the definition).

The algorithm tries to add each block from the list $L_i^*$ to the current partial solution and generate a new partial solution. Evidently, the blocks of a new
partial solution must be disjoint. Moreover, new partial solution must contain the operation \( i^* \). Otherwise, this operation will never have been assigned further and new partial solution cannot be developed to a complete one. All new partial solutions which do not respect these conditions or have the objective which greater or equals than the cost of the current complete solution are eliminated from the search.

The rested partial solution having minimum cost is developed at first, i.e. it becomes the current partial solution and the search process is repeated.

- If the algorithm obtains a new complete solution, then the cost of this solution is compared with the current best cost. If smaller, then the new solution becomes the current solution and a standard backtracking procedure is applied.
- Each node of search tree is developed by the described way. The algorithm continues to process while the search tree has undeveloped nodes.

Although this backtracking algorithm can be directly applied to the problem (9), (12), (13), it is desirable to transform it to a branch and bound algorithm. This transformation increases the efficiency of the search but requires computing of the lower bound of objective \( LW(X) \) for arbitrary set \( X \).

Let \( X^* \) be an optimal solution of the problem (9), (12), (13) and \( X \) be a partial solution such that \( X \subseteq X^* \). Denote \( \widetilde{LW}(X) \) is a lower bound of the objective \( LW(X) \) and \( \mathcal{N}(X) = \mathbb{N} \setminus N(X) \) is a set of unassigned operations.

Clearly, \( \widetilde{LW}(X) \) must be smaller or equal than \( LW(X^*) \). Lower bound \( \widetilde{LW}(X) \) can be written as:

\[
\widetilde{LW}(X) = \sum_{s \in S, s \notin X} c(s) + C_0(m - 1) + \sum_{x \in X} g_x + \sum_{i \in \mathcal{N}(X)} f_i, \tag{14}
\]

where number \( g_x \) is a lower bound of the assignment cost of block \( x, x \in X \) and number \( f_i \) is a lower bound of the assignment cost of operation \( i \in \mathcal{N}(X) \).

The numbers \( g_x, x \in X \) can be defined as:

\[
g_x = c(x) + C_0 \max \left( \frac{t(x)}{T_0}, 1/n_0 \right). \tag{15}
\]

The numbers \( f_i \) can be estimated by the following way. Let \( \Gamma = \{ b \mid b \in \mathbb{B}, \mathcal{N}(b) \subseteq \mathcal{N}(X) \} \) be a set of blocks which do not contain any operation assigned by the blocks of the set \( X \). Due to the fact that the operations are grouped into the blocks and blocks of any feasible \( X \) are disjoint, each unassigned operation \( i \) of the set \( \mathcal{N}(X) \) must belong to only one block \( x \) from the set \( \Gamma \) in the given optimal complete solution \( X^* \). All the operations of block \( x \) are assigned together. Therefore, the average cost of
each operation of block $x$ can be computed as $c(x)/|N(x)|$. Moreover, due to the fact that block $x$ is assigned to a workstation, the cost of each occupied workstation can be separated between the blocks assigned to it. Namely, using the cycle time and block number constraints the lower bound of this number can be computed as $C_0 \max(t(x)/T_0, 1/n_0)/|N(x)|$. Aforementioned estimates lead that the lower bound of the contribution of the operation $i$ to the objective $LW(X)$ equals:

$$\frac{c(x) + C_0 \max(t(x)/T_0, 1/n_0)}{|N(x)|}, \; i \in x.$$  

In general, an operation $i \in \overline{N}(X)$ can belong to several blocks from the set $\Gamma$ and it is not known beforehand what block will belong to the set $X^*$. However, the numbers $f_i$ can be computed as:

$$f_i = \min_{b \in \Gamma, i \in b} \left( \frac{c(b) + C_0 \max(t(b)/T_0, 1/n_0)}{|N(b)|} \right), \; i \in \overline{N}(X). \tag{16}$$

Clearly, if $f_i$ and $g_x$ numbers are computed by formulas (15) and (16), then $\overline{LW}(X) \leq LW(X^*)$, for any $X \subseteq X^*$. This proves the value $\overline{LW}(X)$ is a lower bound of $LW(X)$.

What distinguishes our proposed algorithm from that cited in [20] are the following:

i) algorithm starts the search from set $X = \{s_{m_1}, \ldots, s_{m_q}\}$;

ii) in each node of search tree a lower bound $\overline{LW}(X)$ is computed by formula (14) if not all the operations are selected. Otherwise, $LW(X)$ is computed by formula (13);

iii) deep first search is applied;

iv) when a node is infeasible or unpromising the search is continued from the most promising undeveloped node.

Let $F$ and $F'$ be a family of sets of blocks satisfied (9). Assume $LW_S$ is the best lower bound. A tree-search algorithm to solve problem (9), (12), (13) may be described as follows:

Algorithm 1.

Input:  Sequence $S$;

Output: Lower bound $LW_S$;

Start:  Let family $F = (\{s_{m_1}, \ldots, s_{m_q}\})$, $F' = \emptyset$, $LW_S = \infty$;

Step 1: Select first set $X$ from family $F$, $F = F \setminus (X)$.  

14
Step 2: If set \( X \) satisfies constraints (12) and \( LW(X) < LW_S \) then assume 
\[ LW_S = LW(X). \] Goto Step 5.

Step 3: If \( LW(X) < LW_S \) then compute 
\[ i^* = \min_{i \in N(X)} i. \] For each block \( b \in L_i^* \) 
generate set \( X' = X \cup \{ b \}. \) If set \( X' \) satisfies constraints (9) then assume 
\[ F' = F' \cup (X'). \] Sort subsets \( X' \) of \( F' \) in increasing order of \( LW(X') \).

Step 4: Assume \( F = F' \cup F, F' = \emptyset; \)

Step 5: If family \( F \neq \emptyset \) then goto Step 1 else goto Step 6.

Step 6: The end.

The result of algorithm 1 is the value \( LW_S \). If \( LW_S < \infty \) then the solution of set partitioning problem (9), (12), (13) exists and the value \( LW_S \) is minimal. Otherwise, problem (9), (12), (13) does not have a solution and sequence \( S \) is infeasible.

Computational experiments (see Figure 13 and 14) have shown that the mean gap and the mean time of the lower bound calculation are sufficiently small.

In the next section, branch-and-bound procedure for solving the initial problem (1)-(8) optimally is presented.

5. Branch-and-Bound Algorithm

The algorithm developed in this paper is derived from the standard branch-and-bound algorithm. The main part of the node extension procedure is described below.

Let \( S = \{ s_{11}, \ldots, s_{1q_1}, \ldots, s_{m1}, \ldots, s_{mq_m} \}, s_{ku} \in B \) be a sequence satisfying constraints (2),(3), (5)-(7) and constraint (4) for workstations 1, \ldots, \( m-1 \). Such sequence will be referred to as a feasible partial solution of problem (1)-(8).

Sequence \( S \) corresponds to a node in the search tree, i.e. the blocks of sequence \( S \) are already assigned to \( m \) workstations. First \( m-1 \) workstations are fixed. Consider the last workstation (with index \( m \)). If block \( b \in B \) satisfies constraints:

\[ \mathcal{N}(b) \cap \mathcal{N}(S) = \emptyset, \quad Pr(b) \subseteq \mathcal{N}(S) \cup \mathcal{N}(b), \] (17)

then sequence \( S \) can generate two possible sequences:

i) \( \tilde{S} = \{ s_{11}, \ldots, s_{1q_1}, \ldots, s_{m1}, \ldots, s_{m(q_m+1)} \}, \) \( s_{m(q_m+1)} = b; \)

ii) \( \hat{S} = \{ s_{11}, \ldots, s_{1q_1}, \ldots, s_{m1}, \ldots, s_{mq_m}, s_{(m+1)1}, s_{(m+1)1} = b. \)

Expressions (17) check constraints (2) and precedence constraints (3) for sequences \( \tilde{S} \) and \( \hat{S} \) easier than it is proposed in the model definition (1)-(8).
In the case i) block $b$ is appended to the last occupied workstation of sequence $S$. Cycle time, number of blocks and block exclusion constraints must be checked for the last workstation.

In the case ii) the last occupied workstation is fixed and block $b$ becomes the first block of the new station $m+1$. Sequence $S$ may not respect constraint (4) for the last workstation $m$. If workstation $m$ becomes a penultimate, then operation inclusion constraints for workstation $m$ must be verified (other constraints are verified along sequence $S$ construction process).

A multi-choice search tree is built. Let $FB(S)$ be a set of blocks which satisfies constraint (17). Thus, the extension of the node is the construction of $\tilde{S}$ and $\hat{S}$ sequences from sequence $S$ for each blocks $b \in FB(S)$. All infeasible generated $\tilde{S}$ and $\hat{S}$ sequences are eliminated from the search.

The general rule for pruning unpromising subtrees is applied. Namely, the node corresponding to the sequence $S$ is unpromising and is pruned if $LW_S \geq CR$, where $CR$ is the current record value of the objective (8). A node having minimal value of the lower bound is considered as most promising and it must be branched first.

In addition, some of the generated $\hat{S}$ sequences can be eliminated without computing their lower bound in corresponding nodes as well. Let sequences $\tilde{S}$ and $\hat{S}$ generated for block $b \in FB(S)$ be feasible and both satisfy the constraints (4). Suppose the sequence $\hat{S}$ can be developed to a complete solution. In this complete solution the block $b$ is the first block of the workstation with number $m+1$. This complete solution can be transformed to another one, where the block $b$ is removed from the first position on the workstation $m+1$ and inserted as the last block of the workstation with number $m$. In fact, the second complete solution is derived from the sequence $\tilde{S}$ and its workstation number is less or equals than the workstation number of the first complete solution. Thus, if the aforementioned conditions are respected, then sequence $\tilde{S}$ dominates $\hat{S}$ and the latter can be eliminated from the search.

Moreover, a large number of nodes can be pruned (deleted) using a simple dominance rule. Such an approach is widely used for reducing of the search tree [8]. This procedure will be called in the rest of paper as the operation dominance checking. It can be described as follows:

i) Due to the fact that the set $Y = \mathcal{N}(S)$ can be considered as a state after sequential assignment of elements from $S$, then during the search each state $Y$ is considered as a candidate to be stored in an appropriate data structure $\textbf{DS}$.
(such as list, array, balanced tree, etc.). Suppose \( C_{DS,Y}^* \) is the current minimal value of the objective \( C(S) \) for the state \( Y \).

ii) There are three kinds of processing for each state which can be stored: adding, checking, updating. All these actions can be done if sequence \( S \) satisfies operation inclusion constraint (4). This constraint allows to avoid the situation when an infeasible partial solution dominates feasible partial solution having greater or equal cost than previous one.

iii) Let sequence \( S \) respecting operation inclusion constraint is under consideration and state of \( S \) is equal to \( Y \). If the data structure \( DS \) does not contain the state \( Y \), then, adding processing is realized with the addition of the state \( Y \) and corresponding cost \( C_{DS,Y}^* \) to the data structure \( DS \). The same rule must be applied for checking and updating. Namely, if \( DS \) contains a state \( Y \) and \( C_{DS,Y}^* \leq C(S) \), then, all \( \hat{S} \) (i.e. for any block of \( FB(S) \) ) sequences based on the sequences \( S \) are eliminated from the search. In this case, the sequence \( S \) is called dominated. Otherwise (i.e. \( C(S) < C_{DS,Y}^* \) ), \( C_{DS,Y}^* \) is updated to \( C(S) \) and \( \hat{S} \) sequences must be extended.

Let \( F \) be a family of sequences \( S \) as well as \( F' \) be a family of new generated sequences (obtained during the node extension). The proposed branch-and-bound algorithm is the following:

**Algorithm 2.**

**Input:** Sets \( N, B, N(b), I^o(i), \) graphs \( G^r, G^{BE}, \) number \( c(b), t(b), C_0, T_0, b \in B, i \in N; \)

**Output:** Optimal sequence \( S^* \) and the optimal value \( CR \) of the objective (8);

**Step 1:** Obtain the lower bound of the root node by solving the problem (9), (12), (13) for the \( S = \{ \emptyset \} \). If \( LW_S < \infty \), then assume \( CR = \infty \), \( F = (\{ \emptyset \}) \). Otherwise, goto Step 10;

**Step 2:** If \( F \) is not empty, then assume the current sequence \( S \) equals the first sequence in the family \( F \), \( F = F \setminus \{ S \} \), \( F' = (\emptyset) \). Otherwise, goto Step 10;

**Step 3:** If \( LW_S \geq CR \), then goto Step 2;

**Step 4:** If \( N(S) = N \) and \( S \) satisfies the constraints (4), then assume \( S^* = S \), \( CR = C(S) \) and goto Step 2;

**Step 5:** Compute the set \( FB(S) \). For each block \( b \in FB(S) \) produce \( \tilde{S} \) sequence. Add feasible \( \tilde{S} \) sequences to \( F' \);
Step 6: Check operation dominance rule the sequence $\tilde{S}$. If it is not dominated, then for each block $b \in FB(S)$ build $\hat{S}$ sequence. If $\hat{S}$ is not dominated by the corresponding $\tilde{S}$, then add $\hat{S}$ to the family $F'$;

Step 7: For each sequence from $F'$ obtain its lower bound. Delete unpromising or infeasible sequences;

Step 8: Sort sequences in $F'$ in increasing order of their lower bounds;

Step 9: Assume $F = F' \cup F$ and goto Step 2;

Step 10: The end.

An example of the search tree generated by Algorithm 2 for the instance in Figure 4 is given in Figure 7.

[Insert Figure 7 about here]

In this figure, the nodes are numbered in order of their generation. The search is started from the root node. If both $\tilde{S}$ and $\hat{S}$ sequence are eliminated by the same reasons from the search, then the corresponding nodes are combined into the same record in the figure. It deals with nodes 11, 12 and 17, 18 as well as 19, 20 and 21, 22. In the node with number 14, the algorithm 2 obtains the first complete solution $S = \{1, 2, 3, 4\}$, $q_1 = 2$, $q_2 = 1$, $q_3 = 1$. Furthermore, the search is continued from the node 15, because the search tree has undeveloped nodes. However, the branching does not lead to a complete solution which is better than the solution in the node 14. Hence, this solution is optimal.

6. Experimental results

The example from the previous section shows to what extent the considered TLBP is complex. Even for the instance in the Figure 7 which has 6 operation and 12 blocks the algorithm 2 has examined 22 nodes.

In general, the enumeration can become extremely time-consuming if the number of blocks and operations increases. Therefore, the main goal of the provided experiments is to ascertain that the proposed approach can be applied to the vast majority of instances with the reasonable size. In addition, the experiments are used to determine the dependence of the solution time and other problem parameters. Due to the aforementioned reason, if several algorithms can solve an instance, then this dependence can be useful to forecast what algorithm could be faster.

Due to the fact that TLBP is $NP$-hard, the solution times of instances have large variations. It is typical that it is impossible to estimate solution time for the
each given instance beforehand. So, all conclusions of solution time behavior which depends on instance parameters can be done in the average case.

From another point of view, it is well-known that the solution time of \( NP \)-hard problems depends not only of the number of their variables but on values of these variables as well. TLBP is a complex problem having more variables which can vary in widely. Hence, it possible to study the dependence of solution time only on aggregate parameters and for most expected values of the parameters in real problems.

The situation is more complex for such parameters as the line cycle time, the block and station costs as well as numbers of operation in blocks and block structures. Evidently, they can influence the solution time. Unfortunately, this influence cannot be significantly established by statistical experimentation because it is necessary to solve an enormous number of instances. Moreover, a slight variation of some parameters can significantly change instance properties. For example, this relates to the line cycle time. A slight variation can change optimal line structure very much (this fact follows from the properties of the knapsack problem because the multiple knapsack problem is a partial case of TLBP).

All aforementioned reasons causes the set of interesting problem parameters. Namely, in this paper, four aggregate parameters are under consideration: the number of operations, the number of blocks and the density of precedence graph as well as the density of the block exclusion graph. Two last parameters are defined by the following definition.

Let \( p(G) \) be the ratio between the number of edges (arcs) in the graph (digraph) \( G \) and the number of edges (arcs) in the complete graph (digraph) with the same number of vertices. The same type of characteristics is used both for the precedence graph \( G^r \), block exclusion graph \( G^{BE} \) and operation inclusion graph \( G^o \).

In most real transfer line balancing problem the number of operation is from 40 to 150. However, for large-scaled instances, some blocks or workstations are often predefined by technological reasons. This significantly reduces the size of problem. Therefore, in practice, it can be expected that the number of operations will not be greater than 70 for most instances.

Evidently, the behavior of the problem objective depends on the ratio between the cost of blocks and workstation. Hence, the cost of blocks and workstation are selected in such a way that the average cost of blocks is approximately equal half of the workstation cost.

The maximal number of block per station \( n_0 \) as well as the cycle time \( T_0 \) and block
execution times $t(b), b \in B$ are generated in such a way that the possible number of blocks assigned to the same workstation is from 1 to 5. Usually, these bounds are reasonable for the considered TLBP.

The density of both the precedence and block exclusion graph is supposed to be from 5% to 15%. These bounds are most useful in order to demonstrate the application of the proposed algorithm. This follows from the fact that SALBP becomes easy for solving if the density of the precedence graph increases [25].

The test instances were generated randomly for four values of $|N|, |B|$ and for three values of $p(G^o)$ and $G^{BE}$ (144 series of 20 instances in each series). For the generation of the instances, the following values of parameters are used: $C_0 = 500$, $n_0 = 5$, $T_0 = 130$, $200 \leq c(b) \leq 280$, $15 \leq t(b) \leq 45$. It is supposed that $p(G^o) = 0.01$ and $p(G^r)$ as well as $p(G^{BE})$ is from the set \{0.05, 0.1, 0.15\}.

The approach of the instance generation is the following. First, the partition of all the operations between workstations is generated. The operations assigned to the same workstation are randomly divided between blocks. The precedence relations are randomly generated in such a way that the constructed partition remains feasible. They are added one by one until the density of the precedence graph is smaller than the predefined value. Further, the set of blocks is supplemented with additional blocks of parallel operations. The blocks exclusion constraints are randomly generated in such a way that the constructed partition remains feasible. Exclusion constraints are added one by one until the density of the block exclusion graph is smaller than the given value. The operation inclusion constraints are generated similarly. The block execution times are randomly generated using the predefined values of the problem parameters. This is processed until the first generated partition does not respect to the cycle time constraints. Finally, the block costs are randomly generated. This approach provides that the generated instance has at least one feasible solution.

Tests were carried out on Intel IV with 2.4Ghz and the results are presented in Tables 1-2 and Figure 8-13.

[Insert Table 1 about here]

[Insert Table 2 about here]

The gap shows the fitness of the lower bound and computed as $(C^* - LB_R)/C^*$, where $C^*$ is the optimal value of objective and $LB_R$ is the root lower bound. Table 2 shows that average gap is relatively small. This justifies that the proposed lower bound is appropriate. However, it is well-known, the set partitioning problem is $NP$-
complete and its solution time increases rapidly. This is justified by the Table 2 as well.

Several figures illustrate the influence of the main problem parameters both on the solution time of instances and the time of the lower bound computation.

Figures 8 and 9 show that the mean solution time of instances increases rapidly if the number of operations and blocks increases.

[Insert Figures 8-9 about here]

Figures 10 and 11 show that density of precedence graph \( p(G^r) \) and density of block exclusion graph have a enormous influence on the solution time. As follows from Figure 10, the solution time decreases when \( p(G^r) \) increases because it reduces the size of the search tree of problem (1)-(8). This is because the mean number of new generated nodes per each node of search tree is significantly less for instances having high value of precedence graph density. It should be noted that the precedence graph density \( p(G^r) \) is greater or equal to 0.10 in most real life transfer line balancing problems.

Figure 11 presents the dependance of solution time and values of \( p(G^{BE}) \) on the number of operations. This figure shows that the solution time and \( p(G^{BE}) \) increase simultaneously. The degree of increase grows with the number of operations.

[Insert Figures 10-11 about here]

This behavior of the solution time can be explained by features of the lower bound. As mentioned in section 4, to obtain a lower bound it is necessary to estimate number of required workstations by formulas (10), (11). If the density \( p(G^{BE}) \) increases, then the number of blocks really assigned to workstations decreases (on average). Hence, estimating the required workstation number is provided by the first and second components in formula (10) in most cases. However, the first component is a lower bound of the clique number and it can essentially differ from the real clique number for an arbitrary graph [1]. Thus, in this situation the gap increases. It is clarified by Figure 12 and Table 2. They show that the mean gap and density of block exclusion graph increase simultaneously, because the number of examined nodes in the search tree grows as well. This is because the branch-and-bound algorithm is used to solve the problem.

[Insert Figures 12 about here]
Figure 13 shows that the gap and density of the precedence graph grow simultaneously. So, similarly to the previous case, the number of examined nodes must increase. However, as stated earlier, the increase of the precedence graph density leads to the number of examined nodes is reduced. Figure 10 clarifies that the direct influence of precedence graph density on the size of search tree is essentially greater than the same influence caused via the gap.

[Insert Figures 13 about here]

In general, for the algorithm which based on the branch and bound approach, any lower bound has two main features. Namely, this is the computation time and gap. Figure 14 illustrates the dependence of the time of obtaining the lower bound on the number of operations and blocks. As follows from this figure, the developed lower bound can be rapidly computed for the instances of reasonable size. Table 2 indicates that the mean gap is relatively small. These reasons justify the interest of the proposed lower bound.

However, the computation time of the lower bound increases rapidly if the problem size grows. This follows from the fact that to obtain the lower bound is to solve a special set partitioning problem which is \( NP \)-complete.

Finally, it is possible to estimate the average relative impact of the problem parameter on the solution time. Let's assume that the solution time is a function of the numbers of blocks and operations plus the densities of precedence graph as well as block exclusion graph. In most cases, it is supposed that this function is a simple weighted sum where the weights are unknown parameters. This sum is called the linear regression model. Its unknown parameters can be estimated by the least-squares method.

Due to the fact the TLBP is a \( NP \)-hard problem it is more appropriate to use an exponential model instead of linear. If \( ST \) is a solution time of an instance, then the model is defined as follows:

\[
ST = \exp^{\alpha_0 + \alpha_1|N| + \alpha_2|B| + \alpha_3p(G^r) + \alpha_4p(G^{RE})}.
\]

The above model can be transformed to linear model:

\[
\ln ST = \alpha_0 + \alpha_1|N| + \alpha_2|B| + \alpha_3p(G^r) + \alpha_4p(G^{RE}).
\]

The results of the estimation of the unknown regression parameters are presented in Table 3. In general, the considered regression is significant. Its main parameters
take the following values: coefficient of determination $R^2$ is equal to 0.51578 and $F$-statistics is equal to 767.397.

[Insert Table 3 about here]

The estimated values of the unknown parameters show the mean contributions of corresponding parameters to the instance solution time (see column 'part of impact'). Then, the considered problem parameters can be arranged in order of their declining impact as follows: density of block exclusion graph, number of operations, density of precedence graph and number of blocks. In general, this order depends on the values of parameters which are used for the generation of instances.

In addition, the experiments show that approximately 98.6% of the generated instances were solved in 3 hours. This fact allows us to assume that most of the real instances (having parameter values closed to considered ones in this paper) can be solved in this time.

7. Conclusion

In this paper, a transfer line balancing problem with blocks of parallel operations has been investigated. The set of available blocks is given. The blocks of the same workstation are executed sequentially. As compared to the standard assembly line balancing problem, the investigated problem has many additional constraints and properties. The economic benefit which can be achieved, justifies the search for an optimal solution, even though it is time-consuming.

To solve the problem exactly, an original lower bound based on the set partitioning problem has been developed. The branch-and-bound algorithm has been implemented.

As the computational results show, the proposed lower bound is efficient enough and the transfer line balancing problem with the number of operations smaller than 70 can be solved in three hours on average.

The considered problem takes into account a given cycle time. But often, it is interesting to see how the line cost changes when the line cycle time is modified. The proposed algorithms can be used for different possible values of the line cycle time. For each, the optimal solution can be found. These solutions have their investment cost and productivity (line cycle time). Then, the entire cost, which is function on investment cost and productivity, should be optimized while considering the marketing forecast and other managerial factors.
Further work will be concerned with improvement of the lower bound which could reduce the search tree. For solving large-scale problems, it will be promising to combine fast heuristic techniques for pruning nodes having closed lower bounds. Also, to increase performance of the solution of transfer line problems, various decomposition techniques might be useful.

Acknowledgment

Thanks to Chris Yukna for his help in English.

References


**Table 1. Mean solution time (in seconds)**

| \(p(G^*)\) | \(p(G^{BE})\) | \(|N| = 40\) | \(|N| = 50\) |
|---|---|---|---|
| \(B\) | \(B\) | 
| 70 | 80 | 90 | 100 | 80 | 90 | 100 | 110 |
| 5% | 5% | 4.5 | 5.4 | 9.6 | 14.6 | 159.9 | 34.3 | 28.7 | 76.5 |
| 10% | 11.4 | 9.4 | 12.7 | 17.6 | 271.7 | 17.8 | 115.1 | 337.3 |
| 15% | 12.2 | 22.1 | 31.1 | 68.5 | 192.1 | 137.1 | 99.2 | 171.4 |
| Mean | 9.3 | 12.3 | 17.8 | 33.6 | 207.9 | 63.1 | 81.0 | 195.1 |
| 10% | 1.9 | 1.4 | 3.1 | 2.7 | 15.0 | 40.8 | 10.5 | 30.9 |
| 10% | 2.2 | 2.8 | 3.4 | 5.3 | 18.9 | 65.1 | 56.3 | 78.5 |
| 15% | 4.5 | 7.6 | 5.4 | 14.8 | 245.5 | 115.7 | 293.3 | 656.8 |
| Mean | 2.9 | 3.9 | 4.0 | 7.6 | 93.1 | 73.9 | 120.1 | 255.4 |
| 15% | 1.5 | 1.7 | 3.2 | 3.4 | 3.5 | 8.2 | 6.0 | 12.9 |
| 10% | 1.6 | 1.9 | 2.9 | 4.3 | 8.5 | 12.0 | 14.9 | 39.0 |
| 15% | 2.6 | 4.5 | 5.0 | 6.7 | 13.5 | 36.3 | 32.1 | 34.3 |
| Mean | 1.9 | 2.7 | 3.7 | 4.8 | 8.5 | 18.8 | 17.7 | 28.7 |
| 5% | 5% | 149.2 | 234.5 | 1639.7 | 1179.1 | 743.6 | 1097.5 | 738.8 | 4677.5 |
| 10% | 587.4 | 604.9 | 1204.4 | 4363.1 | 1706.2 | 2507.6 | 8103.8 | 8910.4 |
| 15% | 1062.9 | 972.6 | 2434.7 | 4384.9 | 4881.1 | 4608.4 | 7567.8 | 13295.7 |
| Mean | 599.8 | 604.0 | 1759.6 | 3309.0 | 2443.6 | 2737.8 | 5470.1 | 8887.7 |
| 10% | 58.4 | 57.3 | 151.6 | 259.8 | 83.5 | 184.1 | 444.2 | 427.5 |
| 10% | 285.7 | 411.6 | 379.4 | 1293.9 | 264.4 | 210.8 | 1051.1 | 1008.4 |
| 15% | 359.0 | 355.2 | 767.8 | 1323.1 | 326.5 | 568.3 | 1386.5 | 1213.5 |
| Mean | 234.4 | 274.7 | 432.9 | 959.0 | 224.8 | 321.1 | 960.6 | 883.1 |
| 15% | 16.4 | 27.8 | 52.3 | 93.1 | 29.2 | 95.4 | 225.2 | 287.3 |
| 10% | 44.8 | 84.8 | 167.9 | 175.4 | 57.7 | 345.9 | 587.9 | 767.2 |
| 15% | 57.1 | 121.9 | 246.6 | 228.7 | 103.2 | 423.8 | 327.6 | 989.1 |
| Mean | 39.4 | 78.2 | 155.6 | 165.8 | 63.4 | 288.4 | 380.2 | 681.2 |
Table 2. Mean gap (in percents)

| $p(G^r)$ | $p(G^{BE})$ | $|N| = 40$ | $|N| = 50$ |
|----------|-------------|----------|----------|
|          | $\mathbf{B}$ |          |          |
| 5%       | 5%          | 70       | 80       | 90       | 100      | 80       | 90       | 100      | 110      |
|          | 1.0         | 4.4      | 3.5      | 4.5      | 2.9      | 2.9      | 3.2      | 3.3      |
|          | 4.2         | 3.1      | 4.5      | 5.1      | 4.4      | 2.5      | 3.9      | 3.7      |
|          | 4.6         | 5.7      | 6.8      | 9.2      | 5.3      | 3.4      | 4.9      | 5.5      |
| Mean     | 3.3         | 4.4      | 4.9      | 6.3      | 4.2      | 2.9      | 4.0      | 4.2      |
| 10%      | 5%          | 4.8      | 5.6      | 6.8      | 7.7      | 4.7      | 5.9      | 3.7      | 5.0      |
|          | 8.5         | 7.0      | 8.9      | 10.3     | 4.6      | 6.7      | 6.9      | 7.6      |
|          | 7.8         | 9.0      | 9.1      | 12.3     | 8.3      | 8.3      | 10.3     | 9.3      |
| Mean     | 7.0         | 7.2      | 8.3      | 10.1     | 5.9      | 7.0      | 7.0      | 7.3      |
| 15%      | 5%          | 9.3      | 8.6      | 6.4      | 10.2     | 4.8      | 6.3      | 7.8      | 8.6      |
|          | 9.5         | 9.2      | 10.6     | 12.7     | 7.3      | 10.2     | 10.8     | 11.0     |
|          | 9.8         | 14.8     | 12.8     | 14.3     | 10.2     | 12.7     | 13.9     | 13.0     |
| Mean     | 9.5         | 10.9     | 9.9      | 12.4     | 7.4      | 9.7      | 10.8     | 10.9     |

|          | $|N| = 60$ | $|N| = 70$ |
|----------|----------|----------|
|          | $\mathbf{B}$ |          |          |
| 5%       | 5%       | 90       | 110      | 130      | 150      | 100      | 120      | 140      | 160      |
|          | 0.7       | 2.9      | 4.4      | 5.0      | 2.1      | 2.4      | 3.4      | 5.6      |
|          | 3.1       | 4.8      | 5.3      | 6.8      | 2.5      | 3.2      | 4.9      | 6.1      |
|          | 3.5       | 6.0      | 7.9      | 8.4      | 3.9      | 4.0      | 6.8      | 8.0      |
| Mean     | 2.4       | 4.5      | 5.9      | 6.8      | 2.8      | 3.2      | 5.0      | 6.6      |
| 10%      | 5%       | 4.7      | 6.5      | 6.8      | 8.4      | 4.2      | 5.9      | 8.5      | 8.1      |
|          | 7.1       | 8.7      | 10.3     | 11.9     | 7.9      | 7.4      | 8.9      | 10.1     |
|          | 10.0      | 11.2     | 12.7     | 15.3     | 8.2      | 11.0     | 13.0     | 12.8     |
| Mean     | 7.3       | 8.8      | 9.9      | 11.9     | 6.8      | 8.1      | 10.1     | 10.4     |
| 15%      | 5%       | 5.4      | 8.8      | 11.1     | 12.6     | 7.1      | 7.9      | 8.4      | 9.8      |
|          | 8.7       | 11.0     | 11.5     | 12.2     | 7.0      | 10.8     | 11.8     | 11.8     |
|          | 11.2      | 13.4     | 14.9     | 15.6     | 9.8      | 10.8     | 12.4     | 13.6     |
| Mean     | 8.4       | 11.1     | 12.5     | 13.5     | 8.0      | 9.8      | 10.9     | 11.8     |
Table 3. Summary of regression results on the mean solution time.

<table>
<thead>
<tr>
<th>coefficients</th>
<th>value</th>
<th>standard error</th>
<th>t-statistics</th>
<th>probability</th>
<th>part of impact</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha_0$</td>
<td>-5.9145</td>
<td>0.204501</td>
<td>-28.922</td>
<td>$1.18 \cdot 10^{-161}$</td>
<td></td>
</tr>
<tr>
<td>$\alpha_1$</td>
<td>0.08087</td>
<td>0.004582</td>
<td>17.649</td>
<td>$2.88 \cdot 10^{-66}$</td>
<td>25%</td>
</tr>
<tr>
<td>$\alpha_2$</td>
<td>0.03897</td>
<td>0.002115</td>
<td>18.427</td>
<td>$9.32 \cdot 10^{-72}$</td>
<td>12%</td>
</tr>
<tr>
<td>$\alpha_3$</td>
<td>-0.0757</td>
<td>0.008140</td>
<td>-9.295</td>
<td>$2.82 \cdot 10^{-20}$</td>
<td>24%</td>
</tr>
<tr>
<td>$\alpha_4$</td>
<td>0.12571</td>
<td>0.008126</td>
<td>15.471</td>
<td>$6.39 \cdot 10^{-52}$</td>
<td>39%</td>
</tr>
</tbody>
</table>
Figure 1. An illustration of transfer line
Figure 2. Station composed of two sequentially activated spindle heads (PCI-SCEMM, Saint Etienne, France)
Figure 3. Two possible solutions: a) operations are executed in series with different spindle heads; b) operations are executed simultaneously with one spindle head
b) the block exclusion graph $G^{BE}$

<table>
<thead>
<tr>
<th>$b_u$</th>
<th>$\mathcal{N}(b_u)$</th>
<th>$c(b_u)$</th>
<th>$t(b_u)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b_1$</td>
<td>1,3,5</td>
<td>24.08</td>
<td>17</td>
</tr>
<tr>
<td>$b_2$</td>
<td>4</td>
<td>22.58</td>
<td>8</td>
</tr>
<tr>
<td>$b_3$</td>
<td>2,7</td>
<td>24.54</td>
<td>18</td>
</tr>
<tr>
<td>$b_4$</td>
<td>6</td>
<td>24</td>
<td>10</td>
</tr>
<tr>
<td>$b_5$</td>
<td>1,5</td>
<td>22.04</td>
<td>12</td>
</tr>
<tr>
<td>$b_6$</td>
<td>3,5</td>
<td>19.6</td>
<td>12</td>
</tr>
<tr>
<td>$b_7$</td>
<td>5</td>
<td>20.82</td>
<td>19</td>
</tr>
<tr>
<td>$b_8$</td>
<td>4,5,6</td>
<td>16.49</td>
<td>16</td>
</tr>
<tr>
<td>$b_9$</td>
<td>4,6</td>
<td>21.32</td>
<td>11</td>
</tr>
<tr>
<td>$b_{10}$</td>
<td>5,7</td>
<td>20.07</td>
<td>12</td>
</tr>
<tr>
<td>$b_{11}$</td>
<td>6,7</td>
<td>21.66</td>
<td>11</td>
</tr>
<tr>
<td>$b_{12}$</td>
<td>7</td>
<td>23.36</td>
<td>10</td>
</tr>
</tbody>
</table>

$T_0 = 27$, $n_0 = 4$, $C_0 = 50$.

Optimal solution of this instance which satisfies the constraints (1)-(7) is \{1, 2, 3, 4\}, $q_1 = 2$, $q_2 = 1$, $q_3 = 1$.

Figure 4. An instance of the problem (1)-(8).
Algorithm 2. Branch-and-bound procedure for the initial problem

Algorithm 1. Backtracking procedure to obtain a lower bound on objective function (calculated for each node of B&B tree)

- Estimation of the minimal station number \( m \) for a node of the backtracking search tree

Figure 5. Decomposition of the optimization procedure
a) a subgraph $G^X$ of block exclusion graph

b) a complement $\overline{G}^X$

c) induced subgraphs $G^X_k, k = 1, 2$

Figure 6. Estimating the number of required workstations.
Figure 7. The search tree generated by algorithm 2 for the instance presented in Figure 4.
Figure 8. The mean calculation time (in seconds) as the function of the number of operations.
Figure 9. The mean solution time (in seconds) as the function of the number of blocks.
Figure 10. The mean solution time (in seconds) as the function of the density of the precedence graph.
Figure 11. The mean solution time (in seconds) as the function of the density of the block exclusion graph.
Figure 12. The mean gap (in percents) as the function of the density of the block exclusion graph.
Figure 13. The mean gap (in percents) as the function of the density of the precedence graph.
Figure 14. The mean time for the lower bound calculation (in milliseconds) as the function of the number of operations and blocks.