A branch-and-bound based solution approach for the mixed-model assembly line-balancing problem for minimizing stations and task duplication costs

Yossi Bukchin *, Ithai Rabinowitch

Department of Industrial Engineering, Tel Aviv University, Ramat-Aviv 69978, Tel Aviv, Israel

Received 20 February 2003; accepted 18 January 2005
Available online 25 March 2005

Abstract

A common assumption in the literature on mixed-model assembly line balancing is that a task that is common to multiple models must be assigned to a single station. In this paper, we relax this restriction, and allow a common task to be assigned to different stations for different models. We seek to minimize the sum of costs of the stations and the task duplication. We develop an optimal solution procedure based on a backtracking branch-and-bound algorithm and evaluate its performance via a large set of experiments. A branch-and-bound based heuristic is then developed for solving large-scale problems. The heuristic solutions are compared with a lower bound and experiments show that the heuristic provides much better solutions than those obtained by traditional approaches.

Keywords: Production; Branch-and-bound; Mixed-model assembly line balancing

1. Introduction and literature review

Assembly lines are special flow-line production systems that are typical in the industrial assembly of high volume standardized commodities (Scholl, 1999). During the assembly process the product traverses the assembly line, station by station, while in each workstation a fixed predetermined set of tasks is performed. Each task is an atomic working unit, which usually requires specific machinery and skill. The assembly line design involves the assignment of these tasks into the workstations, subject to given precedence relationships among the tasks.

The least complex configuration of an assembly line is the single-model assembly line. The most common objective of the single model assembly
The line-balancing problem (SALB-P) is to maximize the efficiency of the assembly line by minimizing the required capacity per unit of throughput. This goal can be attained either by minimizing the number of workstations given a required cycle time or by minimizing the cycle time given the number of workstations. Comprehensive surveys of related research appear in Baybars (1986), Ghosh and Gagnon (1989) and Scholl and Becker (in press).

The mixed-model assembly line is a more complex environment in which several variants of the product, referred to as models, are assembled simultaneously on the line. The line-balancing problem in a mixed-model environment (MALB-P) involves the assignment of tasks of all models to the workstations. This problem is much more complex since it entails the additional considerations of the interactions between the assembled models. MALB-P reflects modern assembly lines more realistically, where demand is characterized by high variability and relatively small volume for each model. This phenomenon can be observed in the increased number of models of cars, TVs, computers, VCRs and many other products. This problem has been investigated during the last four decades, where some of the earliest works were those of Thomopoulos (1967, 1970). For recent surveys of the various types of assembly line-balancing problems, see Erel and Sarin (1998) and Becker and Scholl (in press).

Due to the increasing importance of the mixed-model assembly lines in modern industry, several works have addressed the MALB-P in the last few years (e.g. Gokcen and Erel, 1998; Erel and Gokcen, 1999; Merengo et al., 1999; Matanachai and Yano, 2001; Vilarinho and Simaria, 2002; Karabati and Sayin, 2003; Xhao et al., 2004). Most of the research on the subject has adopted an approach in which each task that is common to several models is restricted to a single workstation. Subject to this restriction, the balancing procedure becomes similar to that used for solving the SALB-P, and consists, in general, of balancing a single combined model, which represents the union of all tasks of all the assembled models. Van Zante-de Fokkert and De Kok (1997) distinguish between two approaches in the mixed-model assembly line-balancing literature, the combined precedence diagram approach and the adjusted task processing time approach. In the latter, the average time weighted by the relative frequency of each model is calculated for each task. Both approaches transform the problem into a single model line-balancing problem.

In this paper, we re-examine the traditional approach, and question the necessity of the above restriction. We claim that imposing this restriction without taking into account the associated costs may result in a reduced number of feasible configurations, possibly overlooking more efficient ones.

In our proposed model, the task assignment constraint is relaxed; although a task of a particular model should be assigned to a single station, tasks common to different models may be assigned to different stations. We shall refer to this as task duplication. Some additional costs are associated with task duplication, such as: costs of machinery and tool duplication, costs associated with the increased complexity of line operation (training, learning) and cost of inventory management. Hence, the objective function is modified as well, and the goal is to minimize the total cost associated with the balancing solution instead of the number of stations alone. Two cost elements are considered in the objective function—the station cost that is the cost proportional to the number of stations in the line and the task cost that captures the additional cost of duplication. Examining the task cost, one can see that since each task has to be performed in at least one station, the total task cost of a balancing solution consists of a fixed part (cost of performing all the tasks, each in a single station, namely, without duplication) and a variable part, which is roughly proportional to the amount of duplications. One should note that the relaxation of the assignment constraint has already been applied in Roberts and Villa (1970) and in Bukchin et al. (2002). However, the significant cost of performing the same task in more than one station was not taken into consideration in these papers and, thus, optimality cannot be attained for the more general case with task duplication costs. The former applied the relaxation for all tasks while the latter divided the tasks into two groups, one for which the constraint should hold and the other for which the constraint is
relaxed. Bukchin et al. (2002) assume a zero cost associated with duplicated tasks (of the second group), hence the objective function is to minimize the number of stations subject to a given cycle time.

In the next section the proposed model is presented along with its integer programming formulation. An example is presented also for illustrating the cost tradeoff in the model. An optimal solution procedure based on backtracking branch-and-bound is introduced in Section 3, as well as a wide range of experiments to test the performance of the algorithm for various values of problem parameters. In Section 4 we present a branch-and-bound based heuristic for solving large-scale problems. A set of experiments of large-scale problems then examines the performance of the heuristic via comparison of the obtained solutions to a lower bound. These experiments also provide justification of the new approach via comparing the results to two extreme cases. Summary and discussion are presented in Section 5.

2. Model description

We shall now specify our MALB model which aims to minimize the total cost associated with the design of the assembly line. The model assumptions are specified as follows:

1. Multiple similar models of the same product are assembled on the same line.
2. Task duration is constant but may differ between models.
3. A cycle time is associated with each model type.
4. The duration of a task is not longer than the cycle time of the associated model (no use of stations in parallel).
5. The station time of each model should not exceed the model’s cycle time.
6. Each task should be assigned to exactly one station in each model.

The input to the model includes the following data: number of models; duration of each task in each model; a set of precedence constraints (the set of precedence constraints may be represented for convenience as a directed acyclic graph—a precedence diagram, where each node represents a task and each directed arc represents a precedence relation); cycle time for each model (the inverse of the required production rate of each model); a cost for each station to be opened in the assembly line; and finally, a duplication cost for each task. Note, that the models assembled on the line are different variants of the same product, and therefore, are assumed to be relatively similar. This should be expressed by a similar set of tasks, similar assembly equipment and similar total time required. The latter is highly important in a mixed-model environment in order to avoid blockage and starvation on the line. In case the models are quite different from each other, the line operation may suffer from blockage and starvation resulting in high idle times and reduced throughput rate. In this case, an implementation of the solution in a multi-model environment, in which different models are assembled in small-to-medium batches on the line sequentially, may be considered. This approach, which is not addressed in this paper, will involve rearrangements of the line equipment when product changes occur and possible additional inventory holding costs depending on the batch sizes (Scholl, 1999).

The solution of the above model is a configuration of an assembly line in which all tasks of all models are assigned to the workstations, and the cycle time and precedence constraints for each model are satisfied.

2.1. Problem formulation

The notation used for the formulation of the problem is as follows:

Parameters

\( n \) total number of different assembly tasks
\( m \) number of models to be assembled on the line
\( t_{ij} \) process time of task \( i \) \((i = 1, \ldots, n)\) when performed on model \( j \) \((j = 1, \ldots, m)\)
\( IP_{ij} \) set of immediate predecessors of task \( i \) in model \( j \)
The problem formulation is as follows:

\[ \text{Min } \sum_{j=1}^{n} c_j \text{ required cycle time for model } j \]

\[ \text{SC } \text{station cost—fixed cost associated with each station} \]

\[ \text{TC}_i \text{ task cost—fixed cost associated with each station to which task } i \text{ is assigned} \]

**Decision variables**

\[ z \text{ number of stations to be used in the assembly line} \]

\[ x_{ijk} = \begin{cases} 1 & \text{if task } i \text{ of model } j \text{ is assigned to station } k, \\ 0 & \text{otherwise}. \end{cases} \]

Note that since the number of stations is a decision variable, we use the number of tasks, \( n \), as an upper bound on this value.

\[ \tau_{ik} = \begin{cases} 1 & \text{if task } i \text{ of any of the models is assigned to station } k, \\ 0 & \text{otherwise}. \end{cases} \]

The problem formulation is as follows:

\[ \text{Min } \left\{ \text{SC} \cdot z + \sum_{i=1}^{n} \sum_{k=1}^{n} \text{TC}_i \sum_{j=1}^{n} x_{ijk} \right\} \quad (1) \]

s.t. \[ \sum_{k=1}^{n} x_{ijk} = 1 \quad \forall i,j, \quad (2) \]

\[ \sum_{k=1}^{n} k \cdot x_{ijk} \leq \sum_{l=1}^{n} l \cdot x_{ijkl} \]

\[ \forall j,g,h \quad \text{s.t. } g \in \text{IP}_{hj}, \quad (3) \]

\[ \sum_{i=1}^{n} x_{ijkl} \cdot t_{ij} \leq c_j \quad \forall j,k, \quad (4) \]

\[ z \geq \sum_{k=1}^{n} k \cdot x_{ijk} \quad \forall i,j, \quad (5) \]

\[ \tau_{ik} \geq \frac{1}{m} \cdot \sum_{j=1}^{m} x_{ijk} \quad \forall i,k, \quad (6) \]

\[ x_{ijk}, \tau_{ik} \in \{0, 1\}. \quad (7) \]

The objective function (1) minimizes the total cost (the sum of the station and task costs). Constraint set (2) ensures that every task required by each model is assigned to exactly one station. The precedence relationships among tasks are captured by constraints set (3) which ensures that a task will be assigned to a certain station \( k \) only if its immediate predecessors have all been assigned to that station or to an upstream station. Constraint set (4) restricts the total process time of each model at each station not to exceed the model’s cycle time. Constraint set (5) determines the total number of stations to be used (equal to the greatest station index number of all tasks and models). Constraint set (6) examines whether task \( i \) of any model is assigned to station \( k \). Finally, constraint set (7) defines all decision variables as binary.

In MALB-P a situation may exist in which a task of a certain model that can be assigned to some early station, \( k \), will be delayed to a later station, say, \( k + \Delta \), imposing an idle time or an empty station for this model. We shall refer to such a situation as **gapping**. For instance, a gap of size 2 would occur in the line configuration of a certain model if two adjacent stations were skipped or if one station was left partially empty and the next station in the sequence was skipped. We shall define the line gap (abbreviated **gap**) as the largest gap in the configured line. Gapping is essential for exploring all feasible balancing combinations. This point will be demonstrated in the next section.

The single-model line-balancing problem (SALB-P) is of NP-Hard complexity (Karp, 1972). Since the single-model line-balancing problem is a special case of the mixed-model line-balancing problem discussed here (where the number of models equals one), the latter is NP-Hard as well.

### 2.2. Example

We shall now present a simple example to illustrate the problem and emphasize the importance of incorporating task costs in its solution. To that end, define \( t_{ai} \) as the number of different stations to which task \( i \) is assigned, \( t_{ai} = \sum_{k=1}^{n} t_{ik} \). Consider a product with two models \((m = 2)\). The assembly process of each model consists of a series of assembly operations selected out of a group of five possible tasks \((n = 5)\). The precedence diagrams for both models are presented in Fig. 1. Data concerning the task process times (in time units) as well as TC\( _i \) values (in cost units) are given in Table 1. As
can be seen in Fig. 1, the set of tasks performed in Model II is a subset of the set of tasks performed in Model I. In addition, the tasks common to both models do not necessarily require the same amount of time to perform.

The station cost, SC, is equal to 10 cost units. The required cycle times for Models I and II are 8 and 5 time units, respectively.

In Table 2, three balancing solutions for the example problem are presented. The first solution is just an arbitrary feasible solution. The second solution (USC) is an optimal solution subject to the unique station constraints, namely, each task is forced to be performed at the same station for all models. The third solution (No USC) is an optimal solution for a situation in which the above unique station constraint is relaxed. The table is divided into blocks each representing a station. In every block, the cells under the station column specify, for every model, the tasks assigned to that station. The cells under the slack column indicate, for every model, the percent of idle time remaining in the station.

In the feasible solution, there are four stations in the line \( z = 4 \). Hence, the station cost component should be equal to \( SC \cdot z = 10 \cdot 4 = 40 \). Two tasks are assigned to more than one station. Task 1 is assigned to Station 2 for Model 1 and Station 1 for Model 2, and Task 3 is assigned to Station 4 for Model 1 and Station 2 for Model 2. Therefore, \( ta_1 = ta_3 = 2 \), whereas \( ta_2 = ta_4 = ta_5 = 1 \). Consequently, the total task cost component is \( \sum_{i=1}^{n} TC_i \cdot ta_i = 48 \). The total line cost is the sum of the station cost component and the task cost component: \( 40 + 48 = 88 \).

The next solution, denoted as USC, corresponds to the traditional literature on mixed-model assembly line balancing according to which the assignment of each task is restricted to one unique station for all models. We can see that the optimal solution contains a gapping situation where Station 2 in Model 2 is empty. Since no other station has been skipped or only partially assigned, the line gap is equal to 1. As a matter of fact, in this case, given the above constraint, gapping is essential for attaining feasibility. The new line configuration consists of four stations, so the station cost component remains \( SC = 40 \). The unique station constraint implies that \( ta_i = 1 \ \forall i \). Therefore, the task component is equal to \( \sum_{i=1}^{n} TC_i = 29 \) (a constant part of the cost that could be omitted from the objective function but is left here for comparison). The total configuration cost is now equal to \( 40 + 29 = 69 \), which is

<table>
<thead>
<tr>
<th>Task</th>
<th>Model I</th>
<th>Model II</th>
<th>TC_i</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>–</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>–</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Model I  
Model II

Fig. 1. Precedence diagrams for Models I and II.

<table>
<thead>
<tr>
<th>Solution type</th>
<th>Model</th>
<th>Station 1</th>
<th>Slack (%)</th>
<th>Station 2</th>
<th>Slack (%)</th>
<th>Station 3</th>
<th>Slack (%)</th>
<th>Station 4</th>
<th>Slack (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feasible</td>
<td>I</td>
<td>2</td>
<td>50</td>
<td>I</td>
<td>25</td>
<td>4, 5</td>
<td>25</td>
<td>3</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>II</td>
<td>1</td>
<td>60</td>
<td>3</td>
<td>20</td>
<td>5</td>
<td>40</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimal</td>
<td>I</td>
<td>1</td>
<td>25</td>
<td>2</td>
<td>50</td>
<td>3</td>
<td>25</td>
<td>4, 5</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>II</td>
<td>1</td>
<td>60</td>
<td>–</td>
<td>100</td>
<td>3</td>
<td>20</td>
<td>5</td>
<td>40</td>
</tr>
<tr>
<td>USC</td>
<td>I</td>
<td>1</td>
<td>25</td>
<td>2, 4</td>
<td>0</td>
<td>3, 5</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>II</td>
<td>1, 5</td>
<td>0</td>
<td>–</td>
<td>100</td>
<td>3</td>
<td>20</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimal</td>
<td>I</td>
<td>1</td>
<td>25</td>
<td>2, 4</td>
<td>0</td>
<td>3, 5</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No USC</td>
<td>II</td>
<td>1, 5</td>
<td>0</td>
<td>–</td>
<td>100</td>
<td>3</td>
<td>20</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
a significant improvement over the arbitrary feasible solution presented above. In the third solution (No USC) the unique station constraint is relaxed. This time Task 5 (with a relatively low task cost) is assigned to two different stations (Station 3 for Model I and Station 1 for Model II). There are only three stations in the line and once again a single gap occurs between Stations 1 and 3 in Model 2. Total line cost improves to $30 + 31 = 61$, which is superior to the previous one. One can see that relaxing the unique station constraint may result in reducing the total cost.

3. The branch-and-bound algorithm

We propose a branch-and-bound backtracking, depth-first approach for solving the mixed-model assembly line-balancing problem. Applying this algorithm, a fast feasible solution is generated and its objective function value constitutes an initial upper bound on the total cost. Next, during the search process other solutions are obtained while improving (reducing) the upper bound value. Branches that yield partial solutions with a lower bound larger than or equal to the best upper bound found so far, are fathomed. The algorithm is terminated when all branches have been explored and an optimal solution is found. Key elements of the branch-and-bound algorithm are first introduced. Next, the algorithm steps are explained. Finally, this section concludes with the description of a set of experiments that were conducted for examining the algorithm performance and how it is affected by the problem parameters.

3.1. Key elements of the branch-and-bound algorithm

We shall now describe the key elements of our branch-and-bound algorithm. First, some additional notation is required:

- $\text{UA}_j$: set of tasks not yet assigned in model $j$
- $\text{AM}_i$: set of models for which task $i$ has already been assigned
- $\text{UAM}_i$: set of models for which task $i$ exists and has not yet been assigned

- $\text{os}_j$: index number of the last station to which a task has been assigned for model $j$
- $\text{sl}_j$: time slack of the last station to which a task has been assigned for model $j$
- $\text{s}_{ij}$: station index to which task $i$ of model $j$ has been assigned
- $\text{ms}_{ij}$: lowest station index that can accommodate task $i$ of model $j$
- $\text{ns}_j$: minimum number of additional stations that will necessarily be opened for model $j$ in order to accommodate its remaining unassigned tasks (including task $i$)
- $\text{gl}_{ij}$: limit set upon the maximum gap size for candidate task $i$ of model $j$: $\text{gl}_{ij} = 1 + \frac{n - \text{ns}_j - \text{ms}_{ij}}{\text{C}_0}$ (recall that $n$ is the upper bound for the number of stations in the line)
- $\text{ata}_i$: number of stations to which task $i$ has actually been assigned

$$\text{nta}_i = \begin{cases} 1 & \text{if task } i \text{ will necessarily be assigned to more than one station,} \\ 0 & \text{otherwise.} \end{cases}$$

3.1.1. Branch-and-bound tree structure

In every node of the branch-and-bound tree an additional task, say task $i$, of model $j$ is assigned to some station $k$. At the root of the tree, no task has yet been assigned, hence, $\text{AM}_i = \{\}$ $\forall i$. At the end/bottom of each branch, a complete solution is obtained where all tasks have been assigned to stations for all the models, namely, $\text{UAM}_i = \{\}$ $\forall i$. Each node of the tree represents a partial solution containing a partial assignment of tasks to stations. This partial solution can be further extended by the assignment of a candidate task which should satisfy the following two conditions: (1) it has no predecessors or all of its predecessors have already been assigned to stations; (2) the station slack is larger than or equal to the task duration.

3.1.2. Bound computations

The lower bound, LB, on the total cost consists of two lower bounds; a lower bound on
the cost associated with the number of stations, \( \text{LB}_S \), and a lower bound on the cost associated with task duplication, \( \text{LB}_D \). Each of these lower bounds consists of two parts: the cost which has accumulated so far through the assignments included in the partial solution, and a lower bound on the cost of the remaining assignments needed to complete the line configuration (for a similar approach, see Bukchin and Tzur, 2000). Let task \( p \) of model \( q \) be the current candidate whose lower bound is to be computed. The various bound components of the associated node are presented next.

### 3.1.2.1. Lower bound on the stations cost, \( \text{LB}_S \)

After the assignment of task \( p \) of model \( q \), \( os_q \) is updated according to the station to which task \( p \) was assigned. The number of stations opened in the partial solution is thus given by \( os = \max\{os_j\} \) and the associated actual station cost is consequently equal to \( SC \cdot os \). To calculate the lower bound on the cost associated with the remaining assignments we do the following. First, a lower bound on the value of \( ns_q \) is developed. The lower bound is equal to the maximum value of the following lower bounds on the number of stations, developed for the SALB-P. The first lower bound, \( \text{LB}_1 \), is equal to the remaining assembly time of model \( q \) yet to be assigned divided by the cycle time of model \( q \) and rounded up to the next integer, namely \( ns_q = \lceil \frac{1}{2} \cdot \sum_{i \in \text{UA}_q} f_{iq} \rceil \). The second and third lower bound, \( \text{LB}_2 \) and \( \text{LB}_3 \) were developed by Johnson (1988). These lower bounds are based on existing limitations in assigning tasks of different durations together in one station (e.g., two tasks which are larger than half of the cycle time cannot be assigned to the same stations). Here, \( \text{LB}_2 \) refers to all tasks that are larger than the half of the cycle time and \( \text{LB}_3 \) to the tasks that are larger than one third of the cycle time and smaller than or equal to the half of the cycle time. The next lower bound, \( \text{LB}_4 \), was developed in Berger et al. (1992), and is based on similar principles as \( \text{LB}_2 \) and \( \text{LB}_3 \). This lower bound, however, addresses simultaneously the two sets of tasks considered in \( \text{LB}_2 \) and \( \text{LB}_3 \). The last lower bound considered here, \( \text{LB}_5 \), is proposed in Scholl and Klein (1997). This lower bound iterates between the lower bound on the number of stations and the lower bound on the cycle time. Each time the lower bound on the cycle time is calculated based on a given lower bound on the number of stations. This calculation considers the minimum amount of time that should be assigned to a station based on the relationship between the number of stations (given by the lower bound) and the number of tasks. For example, say that the lower bound on the number of stations is five and that six tasks have to be assigned. It is clear then that the largest task and the sum of durations of the two smallest tasks are both lower bounds on the cycle time. Each time the calculated lower bound on the cycle time is larger than the given cycle time, the lower bound on the number of stations is increased by 1, and the process is repeated. The lower bound on the number of stations that have yet to be opened for model \( q \) is thus \( ns_q = \max_{r=1,\ldots,s}\{\text{LB}_r\} \).

The minimum number of additional stations yet to be opened in the final solution is therefore \( ns = \max\{ns_l\} \). Hence, the lower bound for cost associated with the number of stations is then \( \text{LB}_S = SC \cdot (os + ns) \).

### 3.1.2.2. Lower bound on the task duplication cost, \( \text{LB}_D \)

In order to calculate the task duplication cost, \( at_{ap} \) is updated by examining the set \( \text{AM}_p \). The actual task cost associated with the tasks already assigned equals \( \sum_{i=1}^n TC_i \cdot at_{ai} \). The calculation of the lower bound on the cost associated with the remaining tasks assignment is based on the detection of cases in which future task duplication will be inevitable. In order to identify future task duplications, tasks are scanned in two courses:

**Across models**—models for which the candidate task \( p \) is required but has not yet been assigned \( (l \in \text{UA}_q) \) are examined. If a model \( l \) exists, such that \( os_l > os_q \) or \( ms_{pi} > os_q \), this means that it would not be possible to assign task \( p \) in the future to station \( os_q \) and therefore, task \( p \) will have to be duplicated. In this case, \( nt_{aq} \) is set to the value of 1. Even if more than one model exists that satisfies the above condition, the future duplication is counted only once, since it is possible that task \( p \) be assigned in the future to the same station for these models. Upon the actual assignment of task
across tasks—tasks that have not been assigned yet for model \( q \) ( \( i \in UA_q \) ) are examined. If a task \( i \) exists, such that \( ms_{iq} > \max_{i \in AM} \{ su_i \} \) as a result of the assignment of task \( p \), this means that it would not be possible to assign task \( i \) in the future to any station it has already been assigned to for another model, and therefore it will necessarily be duplicated. In this case, \( nta_i \) is set to the value 1. Upon the actual assignment of a task \( i \) of model \( q \), \( nta_i \) will be reset to 0 and \( atai \) will be incremented by one unit. The lower bound for the costs associated with task duplication is then \( LB_D = \sum_{i=1}^{n} TC_i \cdot (atai + nta_i) \).

### 3.1.2.3. Lower bound summary

The lower bound on the total cost, consisting of the above two lower bounds, is given by \( LB = LB_S + LB_D = SC \cdot (os + ns) + \sum_{i=1}^{n} TC_i \cdot (atai + nta_i) \).

A special case of the LB is the initial lower bound for the entire problem, which shall be denoted as PLB. The station cost component of the PLB is evaluated in the same manner, as any LB station cost component: \( SC \cdot (os + ns) \), only here \( os = 0 \). The task cost component is simply \( \sum_{i=1}^{n} TC_i \) representing a line ideally configured with no task duplication.

### 3.1.3. Gapping

As was demonstrated in the former example (see Section 2.2), gapping is an important and sometimes essential element in creating efficient line configurations. In order to obtain the optimal solution, all possible station gaps must be examined for each task assignment. In other words, as soon as a task is assigned for the first time, all possible stations that can accommodate this task are considered. Consequently, a set of replicas must be produced for each candidate for a range of stations: \( k = ms_{ij} + g \) (\( g = 0, 1, 2, \ldots, gl_{ij} \)). Let us refer to the example problem presented in Section 2.2. Suppose that Task 1 of Model 1 has already been assigned to Station 1 and now Task 2 of the same model is a candidate for assignment. Task 2 would not fit in Station 1 (\( t_{21} = 4 > c_1 - l_1 = 2 \)), therefore, it may only be assigned to stations with an index equal to or larger than 2 (\( ms_{21} = 2 \)). In order to find the station with the largest index to which Task 2 may theoretically be assigned, we calculate \( ns_1 = \max_{i=1}^{n} [LB_i] = 2 \). Hence, the maximum gap size for Task 2 would be \( gl_{21} = 1 + n - ns_1 - ms_{21} = 1 + 5 - 2 - 2 = 2 \). Therefore, three replicas are created for the candidate assignment of Task 2: one for Station 2 (\( k = 2 \)), one for Station 3 (\( k = 3 \)) and one for Station 4 (\( k = 4 \)).

### 3.2. Algorithm description

The branch-and-bound optimal algorithm consists of the following steps:

1. Create a dummy start node (no tasks has yet been assigned) and evaluate PLB. Set an arbitrary (large enough) initial value as the upper bound on the objective function value, UB.
2. Create a primary candidate list (not including gap replicas yet) for the new node consisting of all tasks \( i \) of every model \( j \) for which \( IP_j \cap UA_j = \{ \} \).
3. If the primary candidate list is empty (a leaf has been reached), then go to Step 9. Otherwise proceed to Step 4.
4. For every candidate, calculate \( gl_{ij} = 1 + n - ns_j - ms_{ij} \), and produce a set of replica candidates for assignment to station \( k \), where \( k = ms_{ij} + g \) (\( g = 0, 1, 2, \ldots, gl_{ij} \)).
5. Compute LB for every candidate.
6. Remove all candidates with \( LB \geq UB \) from the list (the branch is fathomed).
7. If candidate list remains empty, go to Step 10. Otherwise, proceed to Step 8.
8. Select the candidate with the lowest LB value and record its assignment as a new node. Return to Step 2.
9. Set UB to the value of the leaf node’s LB (a new interim solution has been found). If UB = PLB, the solution cannot be improved any further, and an optimal solution has been obtained. In this case, terminate. Otherwise proceed to Step 10.
10. Backtrack until: (a) a node with a non-empty candidate list is reached, or (b) the dummy start node is reached and its candidate list is empty. In the first case go to Step 6. In the
second case, Terminate (the tree has been completely enumerated and the last solution found is the optimal solution).

3.3. Experiments on the optimal algorithm

3.3.1. Experimental design

In this section, the performance of the optimal branch-and-bound algorithm is examined via a wide range of experiments. The purpose of these experiments is to examine the algorithm’s performance under various problem parameters and to analyze characteristics of the optimal solution. The algorithm has been implemented using C++ programming language and run on a Pentium III computer with a 600 MHz processor.

3.3.1.1. Problem parameters. A set of problems was constructed using four problem parameters as follows:

1. Number of tasks (n)—Number of tasks in the combined precedence diagram.
2. Number of models (m)—Number of product variants.
3. Mixed-model flexibility measure (MMFM)—An extension to the single-model F-ratio, developed by Mansoor and Yadin (1971). The F-ratio of a given single model reflects the degree of flexibility in sequencing the tasks of that model. This flexibility depends directly on the model’s precedence relations. A single-model’s F-ratio is simply one minus the ratio between the number of existing precedence relations and the maximum theoretical number of precedence relations in that model, \( \frac{n(n-1)}{2} \). Namely, the value of the F-ratio ranges between zero (a single assembly sequence, no flexibility at all) and one (no precedence constraints, maximal flexibility). The MMFM is defined as the average F-ratio over all the different models. Note, that since some of the models may lack certain tasks, a different \( n \) may be employed for each model.
4. Mixed-model similarity measure (MMSM)—A measure developed to weigh the similarity among the models. Two models are considered as similar models if they require similar sets of tasks with similar time duration of these tasks. The similarity of a group of models is defined as the average similarity of all pairs of models in the group. The mixed-model similarity measure is then

\[
MMSM = 1 - \frac{2}{n \cdot m \cdot (m - 1)} \sum_{i=1}^{n} \sum_{j=1}^{m-1} \sum_{k=j+1}^{m} \frac{|t_{ij} - t_{ik}|}{c_j c_k}
\]

3.3.1.2. Experiment responses (performance measures). Our analysis focused on the effect of the above problem parameters on three responses:

1. Algorithm running time in CPU seconds—A measure for the algorithm’s efficiency. Here we examine how the problem size affects the algorithm running time and whether the flexibility and similarity characteristics of the problem have any influence on the solution time.
2. Maximum gap size—Since a large number of candidates is created and evaluated just for testing every possible gap size, it is important to characterize the typical maximum gap size actually needed, and to determine whether problem characteristics have any influence on its size. The results of this examination may prove to be valuable later on in the heuristic development stage.
3. Distance of the optimal solution from PLB—The initial lower bound, PLB, incorporates neither precedence constraints nor cost considerations; therefore, in most cases it is not expected to be equal to the value of the optimal solution. However, in this experiment the quality of this lower bound, which is a partial indication of the effectiveness of the branch-and-bound algorithm, is examined in terms of the distance from the optimal solution.

3.3.1.3. Experiments. A set of problems was constructed with various parameter values. The parameters for problem size were as follows: \( n = 10, 15 \) and 20 and \( m = 3 \) and 5. Assembly lines
are characterized in practice by low flexibility and by high similarity between models. Therefore, values for MMFM and MMSM were selected in the low and medium range for flexibility and in the medium and high range for similarity as follows: 0.2 and 0.5 for MMFM and 0.6 and 0.8 for MMSM.

A full factorial experimentation has been performed. For each experiment, three replications were generated by randomly generating the remaining problem parameters. A total of 72 problems were solved. The task durations were randomly generated within the range of 2–24 time units. The cycle time values for each model \( c_j \) were randomly taken from a uniform distribution \( U(24, 35) \) for problems with MMSM = 0.6, and \( U(36, 45) \) for problems with MMSM = 0.8. The task costs were generated within the range of 0–20 cost units. The station cost was selected to be between 110% and 140% of the largest task cost.

3.3.2. Results

Although 72 problems were examined, an optimal solution was obtained for 69 of them in a reasonable time. The remaining three problems, the 20/5/0.5/0.6 \((n/m/MMFM/MMSM)\) problems were aborted after a run time of 48 hours with no solution obtained and were therefore excluded from the analysis.

The effects of the control parameters on the dependent variables were examined. ANOVA tables for each of the dependent variables are presented in Tables 3–5. After the examination of several transformations the best model for all three dependent variables was obtained using the natural log transformation, namely, by taking \( \ln(\text{dependent variable}) \) as the new dependent variable. This is not surprising considering the exponential nature of the problem.

3.3.2.1. Run time. The complexity of the algorithm is approximately the number of the nodes generated, multiplied by the complexity of the work to be done at each node. The latter, associated with the lower bounds calculations, is \( O[m \cdot n \cdot \log_2(nm)] \), since the bounds are applied for all the precedence diagrams of all model (maximum of \( n \) tasks per model), and some bounds (LB4 and

<table>
<thead>
<tr>
<th>Source</th>
<th>Sum of squares</th>
<th>DF</th>
<th>Mean square</th>
<th>F-value</th>
<th>Prob &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>1463.86</td>
<td>5</td>
<td>292.77</td>
<td>44.90</td>
<td>&lt;0.0001</td>
</tr>
<tr>
<td>Tasks ((n))</td>
<td>1274.55</td>
<td>2</td>
<td>637.27</td>
<td>97.74</td>
<td>&lt;0.0001</td>
</tr>
<tr>
<td>Models ((m))</td>
<td>133.29</td>
<td>1</td>
<td>133.29</td>
<td>20.44</td>
<td>&lt;0.0001</td>
</tr>
<tr>
<td>Flexibility ((MMFM))</td>
<td>12.82</td>
<td>1</td>
<td>12.82</td>
<td>1.97</td>
<td>0.1657</td>
</tr>
<tr>
<td>Similarity ((MMSM))</td>
<td>104.90</td>
<td>1</td>
<td>104.90</td>
<td>16.09</td>
<td>0.0002</td>
</tr>
<tr>
<td>Residual</td>
<td>417.30</td>
<td>64</td>
<td>6.52</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cor total</td>
<td>1881.16</td>
<td>69</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Source</th>
<th>Sum of squares</th>
<th>DF</th>
<th>Mean square</th>
<th>F-value</th>
<th>Prob &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>0.98</td>
<td>5</td>
<td>0.20</td>
<td>3.64</td>
<td>0.0058</td>
</tr>
<tr>
<td>Tasks ((n))</td>
<td>0.14</td>
<td>2</td>
<td>0.07</td>
<td>1.29</td>
<td>0.2836</td>
</tr>
<tr>
<td>Models ((m))</td>
<td>0.00</td>
<td>1</td>
<td>0.00</td>
<td>0.00</td>
<td>0.9462</td>
</tr>
<tr>
<td>Flexibility ((MMFM))</td>
<td>0.19</td>
<td>1</td>
<td>0.19</td>
<td>3.46</td>
<td>0.0675</td>
</tr>
<tr>
<td>Similarity ((MMSM))</td>
<td>0.62</td>
<td>1</td>
<td>0.62</td>
<td>11.55</td>
<td>0.0012</td>
</tr>
<tr>
<td>Residual</td>
<td>3.44</td>
<td>64</td>
<td>0.05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cor total</td>
<td>4.42</td>
<td>69</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
LB₃) require that the tasks are sorted according to decreasing task durations. Due to the exponential nature of the branch-and-bound algorithm, one would expect that the parameters \( n \) and \( m \) (the number of tasks and number of models, respectively), which constitute the size of the problem, would have a significant effect on the run time. As was expected, we can see in Table 3 that both \( n \) and \( m \) affect the run time significantly. More specifically, the average run time obtained for \( n = 10 \), 15 and 20 is 0.3, 143 and 407 minutes respectively. These figures demonstrate the exponential nature of the algorithm. Regarding the number of models, the average run time obtained for \( m = 3 \) and \( 5 \) is 80 and 275 minutes, respectively. In addition, we have found that the run time decreases with the similarity measure (MMSM). No significant effects were obtained with regard to the flexibility measure.

### 3.3.2.2. Largest gap size.

All problems examined had a gap of at least size 1. However, only problems with lower similarity (MMSM = 0.6) had a maximum gap larger than 1. This is clearly seen in Table 4, where similarity has a significant negative effect \((p < 0.01)\) on the maximum gap size. More specifically, the average largest gap size obtained for lower similarity problems is 1.303 while a value of 1.00 was obtained for the higher similarity problems.

In addition, flexibility was also shown to have a somewhat significant negative effect \((p < 0.1)\) on the maximum gap size. Since gapping is quite costly, a more flexible problem will offer more opportunities for gap avoidance, hence the negative effect. The average largest gap obtained for the lower flexibility problems is 1.23 while a value of 1.06 was obtained for higher flexible problems.

### 3.3.2.3. Optimal solution versus the lower bound value.

A 95% confidence interval for the difference between the optimal solution value and the initial lower bound value (PLB) is only \( 5.78 \pm 1.39\% \). The only parameter found to have any significant effect on this value was the number of tasks, as shown in Table 5 \((p < 0.05)\). Since this factor has three levels, we have looked at the detailed results. Surprisingly, we have seen that the value of this factor increases significantly from \( n = 10 \) to 15 (4.8% for the former and 6.8% for the latter). However, the value of this factor slightly decreases for \( n = 20 \) (5.3%). Hence, despite the fact that the model and the main effect of the number of tasks are significant, we would recommend to avoid drawing any harsh conclusions regarding this effect.

To conclude, one can see that the above mentioned models are statistically significant with \( p \)-value lower than 0.05. Yet, there is a large difference between the first model and the other two with regard to the amount of variance explained by the model. In the former (run time), 77.8% of the variance is explained by the model, while in the latter models (largest gap size and optimal solution versus lower bound) only 22.2% and 20.9% of the variance is explained, respectively. Consequently, one could use the first model as a predicting model of the run time, however, the prediction capability of the latter two responses is limited.

### Table 5

ANOVA table for optimal solution versus the lower bound

<table>
<thead>
<tr>
<th>Source</th>
<th>Sum of squares</th>
<th>DF</th>
<th>Mean square</th>
<th>( F )-value</th>
<th>Prob &gt; ( F )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>53.72</td>
<td>5</td>
<td>10.74</td>
<td>3.38</td>
<td>0.0090</td>
</tr>
<tr>
<td>Tasks (( n ))</td>
<td>31.52</td>
<td>2</td>
<td>15.76</td>
<td>4.95</td>
<td>0.0100</td>
</tr>
<tr>
<td>Models (( m ))</td>
<td>9.31</td>
<td>1</td>
<td>9.31</td>
<td>2.93</td>
<td>0.0920</td>
</tr>
<tr>
<td>Flexibility (MMFM)</td>
<td>7.60</td>
<td>1</td>
<td>7.60</td>
<td>2.39</td>
<td>0.1272</td>
</tr>
<tr>
<td>Similarity (MMSM)</td>
<td>5.18</td>
<td>1</td>
<td>5.18</td>
<td>1.63</td>
<td>0.2066</td>
</tr>
<tr>
<td>Residual</td>
<td>203.54</td>
<td>64</td>
<td>3.18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cor total</td>
<td>257.26</td>
<td>69</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4. Branch-and-bound based heuristic

4.1. Determining heuristic rules

Most industrial environments are characterized by large-scale problems. Since the problem, as mentioned above, is NP-hard, a heuristic approach is required. To that end, a branch-and-bound based heuristic, which aims to reduce the search required for optimal solution, is developed here. The main principle here is to cut branches with small probability of providing optimal solutions. To that end, three heuristic rules have been examined, as follows:

4.1.1. Greedy rule

The first heuristic rule we suggest is to terminate the original algorithm as soon as step 9 occurs for the first time (i.e., adopt the first feasible solution which constitutes the initial upper bound). This solution, which we denote as the greedy solution, can be obtained in a very short (polynomial) time. The value of the greedy solution versus the optimal solution was examined via the 45 largest test problems \( n = 15, 20 \) that were solved to optimal in the previous section. A 95% confidence interval for the difference between the greedy solution value and the optimal solution value was \( 20.53 \pm 2.73\% \). None of the problem parameters tested had any significant effect on this value. While extremely fast, this solution is definitely not optimal and in some cases may present a solution highly inferior to the optimal one.

4.1.2. Gap = 1 rule

As we have discussed above, for problems with high similarity (e.g. MMSM = 0.8) experiments show that the gap needed to balance the line does not exceed the size of one. Even with lower similarity, the gap is of size one in most cases and the average maximum gap size for all of the problems we tested was 1.15. Therefore, by constraining step 3 of the algorithm, setting the value of \( gl_{ij} \) to one, a vast number of calculations could be eliminated, cutting down the running time without significantly compromising the quality of the solution (especially for problems with high similarity). Several cases were observed in which unconstrained problems with gaps larger than one were solved optimally with the gap = 1 constraint (this could happen when more than one line configuration is optimal).

4.1.3. Reduced candidate list rule

A rather strong and highly flexible rule is removing less promising candidates from the candidate list of a new node as soon as it is formed. The removal of every such candidate could imply the truncation of an entire sub-tree. We define \( r \) as the proportion of candidates to remain in the reduced list. The reduced list is integrated in Step 5 of the algorithm, where the \( [100 \cdot (1 - r)] \) percent of the candidates in the list (\([x]\) denotes the largest integer smaller than or equal to \( x \)) with the worst LB values are removed. Thus \( r = 1 \) represents a complete candidate list as in the optimal algorithm and \( r = 0 \) represents the greedy heuristic rule described above.

4.1.4. The proposed heuristic algorithm

The gap = 1 rule and the reduced list rule were combined into the heuristic algorithm in order to shrink the size of the candidate lists created each time a node is formed, thus fathoming branches with lower likelihood of attaining optimal solutions. First, at the most, only one replica of each candidate is then created instead of a large series of replicas. Second, the list itself will be diminished to \( 1 - r \) of its original size.

4.2. Experiments for moderate-size problems

In this section, we examine the effect of the heuristic parameter, \( r \), on the tradeoff between the solution’s quality and the run time. To that end, an additional set of experiments on the largest problems which could still be solved by the optimal algorithm \( n = 15, 20 \), was performed. The gap size was restricted to 1 (gap = 1 rule) and \( r \) was set to six different values: 0.0, 0.2, 0.3, 0.4, 0.6 and 0.8.

The results are presented in Table 6. The average ratio between the values of the heuristic and the optimal solution are presented as well as the run time as a percentage of the time required to solve the problem to optimal. In addition, 95%
half-width confidence intervals for these values are presented and show a relatively small variance of the results (each value in the table is the average over 45 experiments).

Except for one case (out of 45), \( r = 0.8 \) yielded optimal solutions in an average 15.4% of the time it took the optimal algorithm to reach that very same solution. Hence further examination of the interval \( r \in [0.8, 1.0] \) was not required.

It is very encouraging to observe that even with candidate lists reduced to only 20% of their original volume, it took about only 1.23% of the time to obtain a solution value within 5% from the optimum.

4.3. Experiments for large-scale problems

Extensive experiments for evaluating the performance of the proposed heuristic for large-scale problems have been performed and are presented next. The first objective of this experimentation is to examine the effectiveness of the heuristic by solving a wide range of problems and comparing the solution values with the problem’s lower bound. The second objective is to justify the proposed approach by comparing the solution values to solution values obtained by using traditional approaches. The traditional approach is represented by two sets of problem instances, which represent two extremes, as follows:

- The first set (Case A). This set contains problems for which no cost is associated with the task duplication, namely; no additional cost is associated with performing common tasks in different stations. This case corresponds to a situation in which each model is balanced separately; namely; the tasks of each model are assigned to stations separately, ignoring the interaction between the different models.

- The second set (Case B). This set contains problems for which the duplication cost is extremely high. This set represents the traditional approach that uses the combined precedence diagram, in which the unique station constraint holds.

Note, that in reality, neither of these extreme cost structures exist, however, the above traditional approaches ignore the actual cost structure (including the stations and the task duplication costs), which is considered in the proposed approach. Hence, the values of the objective functions of the above two sets were recalculated using the real cost values, and were compared to the solutions obtained by the proposed algorithm (Case C).

In the experimentations, 50-task 3-model problems were solved. In order to cover a wide range of problems parameters, a full factorial design has been conducted based on the following three factors:

1. Mixed-model flexibility measure (MMFM)—three values of 0.50, 0.65 and 0.80.
2. Cycle time—the cycle time of each model was set to be a product of a given constant and the average task duration of that model. In this set of experiments, the constant was set to 5 and 7.
3. Cost ratio—in order to examine different values of task duplication cost, two cases were considered in which the ratio of the randomly generated station costs and the average task cost were set to approximately 5 and 10, respectively.

When a certain task is required by several models, we allow the process time to change among models. Still, one would assume that the process
times of this task for the different models will be somewhat similar. Therefore, we first randomly generated the mean process time of each task, \( t_i \), say, \( t_i \). Then, the process time for each model performing task \( i \) was taken from a uniform distribution around the above value, \( U[0.5t_i, 1.5t_i] \). The duplication costs were also randomly generated from a uniform distribution between 1 and 20 cost units.

Three replications have been performed for each experiment, and the task durations and the cost values were randomly generated for each instance. Thus, 36 problems were solved for each case, for a total number of 108 problems. Since these problems could not be solved to optimal, each problem was solved for three different values of \( r \), 0.2, 0.5 and 0.8. The maximum run time for each problem was set to 5 hours, and the best solution obtained up to this time was considered. The results appear in Table 7. In the first column we can see that each problem is denoted as a four-digit number. The first three digits from the left represent the three factors and denote the factors’ values. The fourth digit represents the replication number of this instance. The columns of each problem case contain the number of stations obtained by the solution, the station cost (the cost per station times the number of stations), the task cost (consisting of the constant value of performing all tasks once plus the duplication cost), and the summation of the latter two components, as the total cost.

For examining the effectiveness of the heuristic, the solution values of the 36 solved problems (Case C) were compared to their respective lower bound on the total cost, developed in Section 3.1. In the last column of Table 7 the difference between each solution value and its lower bound, expressed in a percent, is presented. The difference is on the average 7.56%, with a maximum value of 19.15% and a minimum value of 0.91%. Looking at the experiments on moderate-size problems in Section 3.3, we can see that the average difference between the optimal solution and the lower bound is 5.78%. Assuming that the lower bound has not improved with the problem size, we can conjecture that the heuristic algorithm provides solution values around 2% of the optimal solutions. Yet, one may state that this conjecture is somewhat risky. Statistical analysis indicates that the difference between the solution value and the lower bound decreases significantly \((p\text{-value} < 0.05)\) with the cycle time value (an average value of 4.40% and 10.72% for large and small cycle times respectively). This result can be explained by the fact that a good balance is much harder to attain for small cycle times; however, this difficulty is not captured by the lower bound.

In the second part of the experimentation, the solutions obtained by the heuristic (Case C) are compared with the two extreme cases, mentioned above, Cases A and B. It is important to note that the heuristic was used to solve the extreme cases as well. All problems of Case A, which considers a zero duplication cost, were solved to optimality, and the optimal number of stations was obtained. Regarding Case B, one might think that applying the combined precedence diagram may simplify the solution procedure and enable using SALB-P algorithms (as can be seen in Van Zante-de Fokkert and De Kok, 1997). However, the procedure proposed here assures that the task assignment of each model separately will satisfy the cycle time constraints, and hence, the combined precedence diagram, which is usually used for converting the mixed-model into a single model, cannot be applied here.

As can be seen in Table 7, the number of stations obtained for Case A is in most cases smaller than or equal to the value obtained in Case C. However, when calculating the total cost, which includes also the duplication costs, we can see that, on the average, the solution value obtained for Case C is smaller than those of Case A by 16.4% and a maximum value of 34.2% (see Column 14 in Table 7). Statistical analysis indicates that the difference between Cases A and C increases significantly \((p\text{-value} < 0.05)\) with the cycle time and decreases with the cost ratio. The former can be explained by the fact that a good balance is much easier to obtain for large cycle times. Consequently, in most of the solutions, the same number of stations were obtained for both cases (A and C), and thus the saving in the duplication cost was much more significant. In the case of a small cycle time, on the other hand, we can see a clear tradeoff.
<table>
<thead>
<tr>
<th>Prob no.</th>
<th>Case A</th>
<th>Case B</th>
<th>Case C</th>
<th>A–C (%)</th>
<th>B–C (%)</th>
<th>Lower bound (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>11</td>
<td>484</td>
<td>647</td>
<td>1131</td>
<td>13</td>
<td>484</td>
</tr>
<tr>
<td>1112</td>
<td>11</td>
<td>484</td>
<td>686</td>
<td>1170</td>
<td>13</td>
<td>484</td>
</tr>
<tr>
<td>1113</td>
<td>11</td>
<td>517</td>
<td>690</td>
<td>1207</td>
<td>13</td>
<td>517</td>
</tr>
<tr>
<td>1121</td>
<td>11</td>
<td>1067</td>
<td>645</td>
<td>1712</td>
<td>13</td>
<td>1067</td>
</tr>
<tr>
<td>1122</td>
<td>11</td>
<td>1056</td>
<td>793</td>
<td>1849</td>
<td>15</td>
<td>1056</td>
</tr>
<tr>
<td>1123</td>
<td>11</td>
<td>979</td>
<td>642</td>
<td>1621</td>
<td>14</td>
<td>979</td>
</tr>
<tr>
<td>1211</td>
<td>8</td>
<td>376</td>
<td>660</td>
<td>1036</td>
<td>9</td>
<td>376</td>
</tr>
<tr>
<td>1212</td>
<td>8</td>
<td>416</td>
<td>796</td>
<td>1212</td>
<td>10</td>
<td>416</td>
</tr>
<tr>
<td>1213</td>
<td>8</td>
<td>368</td>
<td>818</td>
<td>1186</td>
<td>9</td>
<td>368</td>
</tr>
<tr>
<td>1221</td>
<td>8</td>
<td>792</td>
<td>768</td>
<td>1560</td>
<td>9</td>
<td>792</td>
</tr>
<tr>
<td>1222</td>
<td>8</td>
<td>744</td>
<td>662</td>
<td>1406</td>
<td>9</td>
<td>744</td>
</tr>
<tr>
<td>1223</td>
<td>8</td>
<td>744</td>
<td>683</td>
<td>1427</td>
<td>10</td>
<td>744</td>
</tr>
<tr>
<td>1211</td>
<td>8</td>
<td>484</td>
<td>633</td>
<td>1117</td>
<td>11</td>
<td>484</td>
</tr>
<tr>
<td>1212</td>
<td>12</td>
<td>528</td>
<td>802</td>
<td>1330</td>
<td>13</td>
<td>528</td>
</tr>
<tr>
<td>1213</td>
<td>11</td>
<td>517</td>
<td>807</td>
<td>1324</td>
<td>13</td>
<td>517</td>
</tr>
<tr>
<td>1221</td>
<td>11</td>
<td>1067</td>
<td>837</td>
<td>1904</td>
<td>14</td>
<td>1067</td>
</tr>
<tr>
<td>1222</td>
<td>11</td>
<td>1056</td>
<td>855</td>
<td>1911</td>
<td>12</td>
<td>1056</td>
</tr>
<tr>
<td>1223</td>
<td>11</td>
<td>979</td>
<td>871</td>
<td>1850</td>
<td>13</td>
<td>979</td>
</tr>
<tr>
<td>1221</td>
<td>8</td>
<td>376</td>
<td>794</td>
<td>1170</td>
<td>9</td>
<td>376</td>
</tr>
<tr>
<td>1222</td>
<td>8</td>
<td>416</td>
<td>779</td>
<td>1195</td>
<td>9</td>
<td>416</td>
</tr>
<tr>
<td>1223</td>
<td>8</td>
<td>368</td>
<td>694</td>
<td>1052</td>
<td>10</td>
<td>368</td>
</tr>
<tr>
<td>1221</td>
<td>8</td>
<td>792</td>
<td>775</td>
<td>1567</td>
<td>9</td>
<td>792</td>
</tr>
<tr>
<td>1222</td>
<td>8</td>
<td>744</td>
<td>727</td>
<td>1471</td>
<td>8</td>
<td>744</td>
</tr>
<tr>
<td>1223</td>
<td>8</td>
<td>744</td>
<td>755</td>
<td>1499</td>
<td>9</td>
<td>744</td>
</tr>
<tr>
<td>1311</td>
<td>11</td>
<td>484</td>
<td>739</td>
<td>1223</td>
<td>13</td>
<td>484</td>
</tr>
<tr>
<td>1312</td>
<td>11</td>
<td>517</td>
<td>801</td>
<td>1318</td>
<td>15</td>
<td>517</td>
</tr>
<tr>
<td>1313</td>
<td>11</td>
<td>1067</td>
<td>785</td>
<td>1852</td>
<td>14</td>
<td>1067</td>
</tr>
<tr>
<td>1321</td>
<td>8</td>
<td>376</td>
<td>698</td>
<td>1074</td>
<td>9</td>
<td>376</td>
</tr>
<tr>
<td>1322</td>
<td>8</td>
<td>416</td>
<td>792</td>
<td>1208</td>
<td>8</td>
<td>416</td>
</tr>
<tr>
<td>1323</td>
<td>8</td>
<td>368</td>
<td>789</td>
<td>1157</td>
<td>9</td>
<td>368</td>
</tr>
<tr>
<td>1321</td>
<td>8</td>
<td>792</td>
<td>756</td>
<td>1548</td>
<td>9</td>
<td>792</td>
</tr>
<tr>
<td>1322</td>
<td>8</td>
<td>744</td>
<td>737</td>
<td>1481</td>
<td>9</td>
<td>744</td>
</tr>
<tr>
<td>1323</td>
<td>8</td>
<td>744</td>
<td>662</td>
<td>1406</td>
<td>9</td>
<td>744</td>
</tr>
</tbody>
</table>

Average = 16.4 8.5 7.56
Max = 34.2 20.4 19.15
between the number of stations (which is smaller in general in Case A) and the duplication cost, which reduces the difference between the results of the two cases.

The next comparison concerns the other extreme in which duplication cost is extremely high (Case B). In this case, which is associated with most traditional models (especially those that use the combined precedence diagram), we can see an average difference value of 8.5% in favor of the proposed model, and a maximal value of 20.4%. In three out of the 36 problems, better results were obtained for Case B, however the differences are relatively small and may be explained by the stochastic nature of the heuristic.

To conclude, we can say that the latter part of the large-scale problems experimentation, the comparison between the proposed approach and traditional approaches, provides justification for the proposed approach and emphasizes its superiority over the traditional approaches. The former part (the comparison between the heuristic results and the lower bound values), in addition, demonstrates the effectiveness of the proposed heuristic while showing that close-to-optimal solutions are obtained.

5. Summary and discussion

A new approach to the MALB-P has been presented in which, in addition to the traditional objective of the minimum number of stations, a task cost is incorporated. The task cost reflects the expenditures spent on providing a selected workstation with the equipment and worker skill needed for its performance at that station. This cost component, as well as the cost involved in setting up and running a workstation per se, are weighed one against the other in the process of determining whether a given task should be allocated to just one station for all models or whether it could be carried out in more than just one station and thus possibly reducing the number of stations required in the assembly line.

An algorithm based on the branch-and-bound technique has been proposed and tested through a series of experiments. In addition, several heuristic rules were developed and a heuristic version of the algorithm has been presented. Experiments for large-scale problems demonstrate the high effectiveness of the procedure and a complementary part of the experiments provide justification for preferring the proposed approach over existing methods.

We believe that our model represents more realistically the set of considerations involved in the design of a mixed-model assembly line and that the algorithms proposed here can efficiently find optimal or near-optimal solutions for this problem. Further research may deal with improving the solution approach for the proposed model, possibly by using meta-heuristics to solve large-scale problems. Another possible direction may involve the extension of the current model by incorporating other real-world features such as the equipment selection and tooling in the balancing process.

References