IIE Transactions (2000) **32**, 585–598

Design of flexible assembly line to minimize equipment cost

JOSEPH BUKCHIN and MICHAL TZUR

Department of Industrial Engineering, Faculty of Engineering, Tel-Aviv University, Tel-Aviv 69978, Israel
E-mail: bukchin@eng.tau.ac.il or tzur@eng.tau.ac.il

Received April 1997 and accepted July 1999

In this paper we develop an optimal and a heuristic algorithm for the problem of designing a flexible assembly line when several equipment alternatives are available. The design problem addresses the questions of selecting the equipment and assigning tasks to workstations, when precedence constraints exist among tasks. The objective is to minimize total equipment costs, given a pre-determined cycle time (derived from the required production rate). We develop an exact branch and bound algorithm which is capable of solving practical problems of moderate size. The algorithm's efficiency is enhanced due to the development of good lower bounds, as well as the use of some dominance rules to reduce the size of the branch and bound tree. We also suggest the use of a branch-and-bound-based heuristic procedure for large problems, and analyze the design and performance of this heuristic.

1. Introduction and literature review

Assembly lines are often used in the last step of production, when the final assembly of the product from previously made parts is performed. An assembly line typically consists of several workstations, each of them being responsible for performing a specific set of tasks. The product moves through the line, from one workstation to the next, according to their order.

The separation of the entire set of tasks into subsets, each performed in a specific workstation, allows for specialization at each workstation. The tasks may be performed manually, or by a dedicated equipment, to achieve high efficiency. Recently, the use of Flexible Assembly Systems (FAS) has been developed, that is, the use of flexible (and usually automated) equipment such as robots or flexible machines, to perform assembly tasks. This development is a particular result of the fast-changing demands of customers which leads to a shorter life cycle of products.

When flexible equipment is used for assembly tasks, the issue of *designing* an assembly line becomes very important. The design in this context consists of selecting the equipment for the workstations, and addressing the related question of which tasks should be performed in which of the workstations. Due to the flexibility of the equipment, there are usually several equipment alternatives for each task, and it may be the case that a particular piece of equipment is efficient for some tasks, but not for others. This has to be taken into consideration when grouping several tasks to be performed at the same workstation, using the same equipment.

In this paper we address the questions of selecting the equipment (flexible assembly machines) and assigning tasks to workstations, when precedence constraints exist among tasks. The solution consists of a series of workstations, where a single specific piece of equipment is placed in each station, and a set of tasks assigned to this station is to be performed by the selected equipment. The objective is to minimize total equipment costs, given a pre-determined cycle time (derived from the required production rate). We develop an exact branch and bound algorithm which is capable of solving practical problems of moderate size. The algorithm's efficiency is enhanced due to the good lower bounds that we develop, and due to the dominance rules that we use to reduce the size of the branch and bound tree. We also suggest the use of a branch-and-bound-based heuristic procedure for large problems, and analyze the design and performance of this heuristic.

As mentioned above, the design problem is to choose the equipment type and set of tasks to be performed in each workstation, and this in turn determines the amount of time it will take to complete all tasks in all the workstations. However, the *balance* amongst all the workstations is very important in the determination of the line's efficiency, and is the subject of a large stream of research. Most of the research performed on the balancing problem deals with *Simple Assembly Line Balancing* (SALB), [1–4] in which no alternative equipment types are considered. That is, every task's time is fixed, and the remaining problem is to determine the sets of tasks to be performed at each workstation. This is clearly a special case of our problem, in which all equipment types are identical. The

0740-817X © 2000 "IIE"

SALB is proven to be an NP-Hard problem [5], and resulting from this is the conclusion that our problem is also NP-Hard.

There are relatively few studies that address the problem in which there is more than one type of equipment. Graves and Holmes Redfield [6] consider the design problem with several equipment alternatives, when multi products are assembled on the same line. They assume a complete ordering of tasks of the same product and large similarities among different products. These assumptions result in a relatively small number of candidate workstations (a number which is proportional to N^k where N is the total number of tasks and k is close to, but greater than two), which therefore simplifies the problem considerably. Their algorithm indeed enumerates all feasible workstations, selects the best equipment for each, and then chooses the best set of workstations. Previous work on the single product design problem with equipment selection includes Graves and Whitney [7] and Graves and Lamar [8], but in both articles the sequence of tasks is also assumed to be fixed.

Pinto *et al.* [9] discuss processing alternatives in a manual assembly line as an extension of SALB. Each processing alternative is related to a given set of tasks i.e., represents a limited equipment selection which may be *added* to the existing equipment in the station, and the decision is whether to use each such alternative in order to shorten the tasks duration, at a given cost. Since the line is manual, each task may be performed at each station. Their solution procedure consists of a branch and bound algorithm in which a SALB problem is solved in every node of the branch and bound tree, therefore this algorithm may be used only for a small number of possible processing alternatives.

Rubinovitz and Bukchin [10] present a branch and bound algorithm for the problem of designing and balancing a robotic assembly line when several robot types are available and the objective is to minimize the number of workstations. Their model is a special case of ours, in which all of the equipment alternatives have identical purchasing costs. Tsai and Yao [11] proposed a heuristic approach for the design of a flexible robotic assembly line which produces a family of products. Given the work to be done in each station, the demand of each product and a budget constraint, the heuristic determines the robot type and number of robots required in each workstation. Their objective is to minimize the standard deviation of output rates of all workstations, which is their measurement for a balanced line.

The remainder of the paper is organized as follows: in Section 2 we introduce the notation and a formulation of the problem and illustrate it with an example. In Section 3 we develop two types of lower bounds, that are used later in our algorithm. In Section 4 we describe our exact branch and bound algorithm and present some qualitative insights with respect to the problem's parameters, based on an empirical study that we performed. We also examine the quality of the lower bound that we developed for the problem. In Section 5 we discuss how a heuristic procedure, based on the branch and bound algorithm, may be designed for the very large problems. Finally, Section 6 contains our conclusions.

2. Problem formulation

In this section we introduce the notation as well as our precise assumptions, and present an integer programming formulation of the problem. Based on this formulation we develop, in the next section, lower bounds for the problem. To illustrate the model's assumptions and help the reader follow our analysis, we provide at the end of this section an example problem.

The problem is defined by the following parameters:

t_{ij} = duration of task i when performed by equipment j $(i = 1, \ldots, n,\ j = 1, \ldots, r)$;
EC_j = cost of equipment type j $(j = 1, \ldots, r)$; (We use interchangeably *equipment* and *equipment type*, this should cause no confusion.)
C = required cycle time;
P_i = set of immediate predecessors of task i $(i = 1, \ldots, n)$.

The following assumptions are stated to clarify the setting in which the problem arises:

(i) There is a given set of equipment types, each type is associated with a specific cost. The equipment cost is assumed to include the purchasing and operational cost of using the equipment.
(ii) The precedence relation between assembly tasks is known.
(iii) The assembly tasks cannot be further subdivided.
(iv) The duration of a task is deterministic, but depends on the equipment selected to perform the task.
(v) A task can be performed at any station of the assembly line, provided that the equipment selected for this station is capable of performing the task, and that precedence relations are satisfied.
(vi) The total duration time of tasks that are assigned to a given station should not exceed the predetermined cycle time.
(vii) A single equipment is assigned to each station on the line.
(viii) A single product is assembled on the line.
(ix) Material handling, loading and unloading times are negligible or included in the tasks duration.
(x) Set up and tool changing times are negligible or included in the task's duration.

The decisions that have to be made address two issues: (i) the design issue, where the equipment has to be selected and assigned to stations; and (ii) the assignment of all tasks to the stations, such that the precedence as well as the cycle time constraints are satisfied. The following two sets of binary decision variables correspond to each of these two issues, respectively. In the Appendix we summarize all the notation used throughout the paper.

We define for every equipment j and every station number k:

$$y_{jk} = \begin{cases} 1 & \text{if equipment } j \text{ is assigned to station } k, \\ 0 & \text{otherwise.} \end{cases}$$

In addition, we define for every task i, every equipment j and station number k:

$$x_{ijk} = \begin{cases} 1 & \text{if task } i \text{ is performed by equipment } j \text{ at station } k, \\ 0 & \text{otherwise.} \end{cases}$$

The following is the resulting integer programming formulation of the problem, denoted as (P1):

$$\text{Min} \sum_{j=1}^{r} \sum_{k=1}^{n} EC_j y_{jk}, \tag{1}$$

subject to

$$\sum_{j=1}^{r} \sum_{k=1}^{n} k \cdot x_{gjk} \leq \sum_{j=1}^{r} \sum_{l=1}^{n} l \cdot x_{hjl} \quad \forall g, h \text{ subject to } g \in P_h, \tag{2}$$

$$\sum_{j=1}^{r} \sum_{k=1}^{n} x_{ijk} = 1 \quad \forall i, \tag{3}$$

$$\sum_{i=1}^{n} t_{ij} x_{ijk} \leq C \cdot y_{jk} \quad \forall j, k, \tag{4}$$

$$\sum_{j=1}^{r} y_{jk} \leq 1 \quad \forall k, \tag{5}$$

$$x_{ijk} = 0,1 \quad \forall i, j, k, \tag{6}$$

$$y_{jk} = 0,1 \quad \forall j, k. \tag{7}$$

The objective function (1) represents the total design cost to be minimized. Note that the number of tasks, n, serves as an upper bound for the number of stations. Constraint set (2) ensures that if task g is an immediate predecessor of task h, then it cannot be assigned to a station with a higher index than the station to which task h is assigned. Constraint set (3) ensures that each task is performed exactly once. Constraint set (4) represents the relationship between the x_{ijk} and the y_{jk} variables by not allowing any task to be performed on a given piece of equipment in a given station, if this equipment is not assigned to that station. Also, if a given piece of equipment *is* assigned to a given station, constraint set (4) specifies the cycle time requirement. Constraint set (5) represents the requirement of at most one piece of equipment at any station and constraint sets (6) and (7) define the decision variables to be binary. Since this is the first time that this problem has been considered, the formulation is new, although elements of it have appeared previously in the literature. The formulation consists of $O(n^2 r)$ variables and $O(n(n + r))$ constraints, but the main importance of this formulation is the relaxation resulting from it, which enables us to obtain good bounds, as explained in Section 3.

2.1. *An example problem*

Our example problem is based on the example analyzed in Pinto *et al.* [9]. In particular, we adopted the precedence diagram of their 10 tasks problem, shown in Fig. 1. In our example, a product is assembled on an automated assembly line, using three different types of equipment (machines). The cost of each equipment type and the time required to perform every assembly task by each of the selected equipment types are shown in Fig. 1. Empty elements in the duration table imply that the task cannot be performed by the associated equipment type.

We can compare among different equipment types along three dimensions: cost, speed and flexibility (number of tasks that can be performed by the equipment). When no equipment type is dominated by the others, a trade off exists between different types, with respect to at least two of the above-mentioned properties. For example: a fast and flexible equipment type is likely to be more expensive. In our example, one can observe that each equipment type has an advantage over the others in one of the three dimensions:

(i) E_1 – a highly flexible equipment type, namely, a piece of equipment which is capable of performing a large number of assembly tasks (all tasks, in this example).
(ii) E_2 – a fast assembly equipment type characterized by short task duration.
(iii) E_3 – the least expensive assembly equipment.

The IP formulation of this example, based on formulation (P1) presented in Section 2, consists of 330 binary variables and 61 constraints.

We solved this problem by our *optimal* algorithm (described in Section 4), determining task assignments and equipment selection, while minimizing the total equipment cost (1), subject to a cycle time constraint of 50. The optimal configuration was obtained in 0.05 seconds and is shown in Fig. 2. The minimal equipment cost required for a cycle time of 50 is \$360 000 (three machines of \$100 000 each plus one machine of \$60 000). The trade off between the different types is demonstrated in the optimal solution, by the fact that all three types of equipment are used. An important conclusion drawn from this example is that as long as a given equipment type is not dominated by another type along *all three dimensions*, it may be included in the optimal configuration.

Equipment	E_1	E_2	E_3
EC_j *($)*	100 000	100 000	60 000
Task 1	8	6	
Task 2	13		14
Task 3	49	40	
Task 4	15		17
Task 5	18	14	
Task 6	15	12	20
Task 7	10	8	
Task 8	10	8	13
Task 9	33		38
Task 10	25	20	28

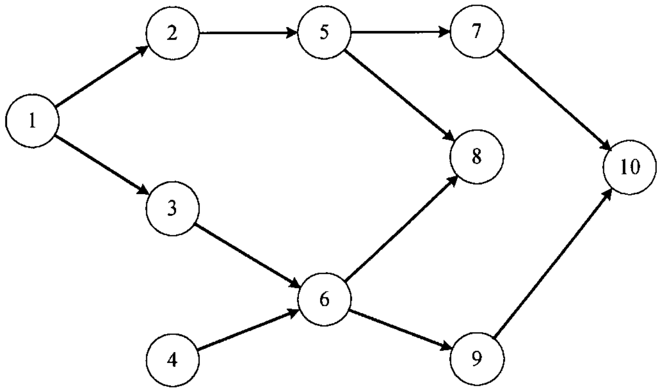

Fig. 1. Precedence diagram, task times and equipment costs.

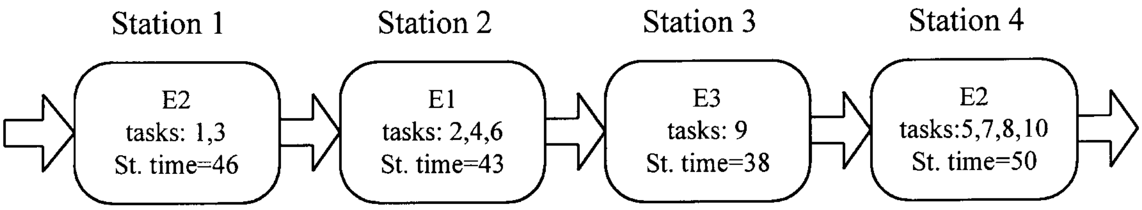

Fig. 2. Optimal configuration of the example problem (total cost = 360 000).

3. Lower bounds

In this section we develop lower bounds for the problem, as well as for subproblems of it. As we show below, the bounds are obtained by relaxing some of the constraints of the formulation (P1) and solving the relaxed problem; the bounds are used in our branch and bound algorithm that will be discussed in the next section.

Consider problem (P1), and make the following relaxations to it:

(i) Eliminate the precedence constraints (2).
(ii) Sum the constraints in (4) over all stations, for each equipment type j. The resulting set of constraints, denoted by (4′) is the following:

$$\sum_{k=1}^{n}\sum_{i=1}^{n} t_{ij}x_{ijk} \leq C\sum_{k=1}^{n} y_{jk} \quad \forall j. \tag{4′}$$

As a result of these relaxations, constraint set (5) is no longer meaningful since all the stations are now considered *together* in the formulation. Equation (4′) implies that it is not required to keep the cycle time constraint in every station, only the *aggregate* cycle time constraint, representing a capacity constraint for each equipment type. Therefore we define the following new decision variables, which are independent of the stations:

$$y_j = \sum_{k=1}^{n} y_{jk} = \text{total number of type } j \text{ equipment.}$$

$$x_{ij} = \sum_{k=1}^{n} x_{ijk} = \begin{cases} 1 & \text{if task } i \text{ is performed by equipment } j, \\ 0 & \text{otherwise.} \end{cases}$$

The relaxed formulation, denoted as (P2), is now described as follows:

$$\text{Min} \sum_{j=1}^{r} EC_j y_j \tag{8}$$

subject to

$$\sum_{j=1}^{r} x_{ij} = 1 \quad \forall i, \tag{9}$$

$$\sum_{i=1}^{n} t_{ij}x_{ij} \leq C \cdot y_j \quad \forall j, \tag{10}$$

$$x_{ij} = 0, 1 \quad \forall i, j, \tag{11}$$

$$y_j \text{ integer} \quad \forall j. \tag{12}$$

Here, (8) is equivalent to (1), representing the total equipment cost to be minimized; (9) replaces (3), ensuring that each task is performed exactly once, and (10) is in fact constraint (4′) discussed above. To simplify the problem further, we relax the *integrality* constraints regarding the y_j variables (12) and obtain problem (P3). Therefore, problem (P3) is defined by (8)–(11) and:

$$y_j \geq 0 \quad \forall j, \tag{12'}$$

and we prove the following:

Theorem 1. *The following solution is optimal for problem* (P3):

$$x_{ij} = \begin{cases} 1 & \text{if } EC_j \cdot t_{ij} = \min_l \{EC_l \cdot t_{il}\} \\ 0 & \text{otherwise} \end{cases} \quad \forall i \tag{13}$$

(*If more than one index j achieves the minimum, choose one of them, arbitrarily.*)

$$y_j = \sum_{i=1}^{n} t_{ij} x_{ij} / C \quad \forall j. \tag{14}$$

Before proving the theorem formally, let us first explain it intuitively. Note first that each unit of an equipment type may be assigned as much work as the cycle time, C. Therefore, the number of units (which may be fractional) to be purchased from each equipment type is the sum of the duration of all tasks assigned to this type, divided by the cycle time, resulting in (14). This means that in order to perform a certain task, say i, by a certain equipment type, say j, a fraction of the equipment needs to be purchased, which equals to the fraction of cycle time required to perform it, i.e., t_{ij}/C. The cost of this fraction of equipment is: $EC_j \cdot (t_{ij}/C)$. Comparing the costs of all equipment alternatives for a given task i, and choosing the type whose cost is minimal, one obtains Equation (13). We now provide a more formal proof.

Proof of Theorem 1. Note first that the solution defined by (13) and (14) is feasible. Note also that given *any* solution to the x_{ij} variables, the solution to the y_j variables as defined by (14) is *optimal*. Therefore it remains to prove that the solution of the x_{ij} variables as defined by (13) is optimal.

Assume, by contradiction, that this solution is not optimal, therefore there exists a variable x_{im} in the optimal solution s.t. $x_{im} = 1$ but

$$EC_m \cdot t_{im} > \min_l \{EC_l \cdot t_{il}\}.$$

This variable, associated with task i, contributes t_{im}/C units to the variable y_m and therefore $EC_m \cdot t_{im}/C$ to the objective value of (P3). If instead we choose for task i $x_{ij} = 1$ for j that satisfies

$$EC_j \cdot t_{ij} = \min_l \{EC_l \cdot t_{il}\},$$

then the contribution to the y_j variable is t_{ij}/C units and therefore $EC_j \cdot t_{ij}/C$ ($< EC_m t_{im}/C$, by definition) to the objective value of (P3), a contradiction to the optimality of the former solution. ■

We define:

$$LB1 = \sum_{j=1}^{r} EC_j y_j, \tag{15}$$

where y_j is determined by (13) and (14).

Corollary 1. *LB*1 *is a lower bound to the value of* (P1).

The Corollary is true since $LB1$ is the optimal objective value of problem (P3) which is a relaxation of problem (P1). In conclusion, we have shown how to obtain a lower bound to the problem, which is easy to compute. The deviation of this bound from the optimal solution value results from ignoring the precedence constraints, from considering the cycle time requirement *in aggregation* to all stations (i.e., a task may be performed in more than one station), and from the ability to use a fraction of a piece of equipment. In the next section a branch and bound algorithm is developed, which uses the proposed lower bound.

In the process of solving problem (P1) via the branch and bound algorithm, a node in the branch and bound tree represents a partial solution, in which some of the tasks have already been assigned to specific equipment types. For this node, the calculation of a lower bound is required. This leads us to consider a *subproblem* of the relaxed problem (P3), in which it is given that a subset of the original set of tasks is performed by an already determined set of equipment types. In addition, a given number of time units, say S, are still available on the last selected equipment type, say type m. Since this equipment has already been purchased, no cost is associated with these S time units. The subproblem has to determine the number and type of *additional* equipment to be purchased (at minimal cost) in order to perform the *remaining* subset of tasks, say σ, and to assign the tasks in σ to the new equipment while satisfying the aggregate (and possibly fractional) cycle time constraint.

We denote this subproblem as (P4) and state its exact formulation:

$$\text{Min} \sum_{j=1}^{r} EC_j y_j,$$

subject to

$$\sum_{j=1}^{r} x_{ij} = 1 \quad \forall i \in \sigma,$$

$$\sum_{i\in\sigma} t_{ij}x_{ij} \le C\cdot y_j \quad \forall j \ne m, \tag{10a}$$

$$\sum_{i\in\sigma} t_{im}x_{im} \le S + C\cdot y_m, \tag{10b}$$

$$x_{ij} = 0,1 \quad \forall i\in\sigma, \forall j,$$

$$y_j \ge 0 \quad \forall j.$$

As discussed, this formulation is identical to (P3), except that only tasks in the set σ are considered, and the original constraint (10) is replaced by (10a) and (10b). Constraint (10b) is a modification of the original constraint (10) for equipment type m, which reflects the S free time units on this equipment type. Ideally, *all* S (free) time units of equipment m should be used, in which case the desired value for the x_{ij} variables may be fractional. Therefore we denote by (P5) a subproblem which is a relaxation of problem (P4), obtained by allowing the x_{ij} variables to be fractional. This relaxation enables us to solve problem (P5) to optimality, providing us with a lower bound to the value of problem (P4). (As becomes clear from the algorithm below, at most two x_{ij} variables, which refer to the same task, will be fractional).

We use the following algorithm, denoted as **Algorithm TES** (Task Equipment Selection), to solve problem (P5):

Step 1. Let $j_{(i)}$ be the equipment type for which $t_{ij_{(i)}}\cdot EC_{j_{(i)}} = \min_l\{t_{il}\cdot EC_l\} \equiv a_i \quad \forall i\in\sigma$

Step 2. Let i^* be the task for which $a_{i^*}/t_{i^*m} = \max_{i\in\sigma}\{a_i/t_{im}\}$

Step 3. If $t_{i^*m} < S$ then: set $x_{i^*m} = 1$, $S = S - t_{i^*m}$ and $\sigma = \sigma\backslash\{i^*\}$. If $\sigma = \Phi$, go to (Step 5); otherwise, go back to (Step 2).

Otherwise: if $m = j_{(i^*)}$ then $x_{ij_{(i^*)}} = 1$,
if $m \ne j_{(i^*)}$ then $x_{i^*m} = S/t_{i^*m}$,
$x_{i^*j_{(i^*)}} = 1 - x_{i^*m}$

$\sigma = \sigma\backslash\{i^*\}$

Step 4. For every $i\in\sigma$ set $x_{ij_{(i)}} = 1$.

Step 5. $y_j = \sum_{i\in\sigma} t_{ij}x_{ij}/C$.

The basic idea of this algorithm is to first assign tasks to the S free time units of equipment type m. Recall that when no free time units are available (as in problem (P3)), every task i is assigned to the equipment which has the minimal value of $t_{ij}\cdot EC_j$ which we define here (Step (1) of Algorithm TES) as equipment type $j_{(i)}$. This is also the solution for problem (P5), once the S free time units of equipment m have been exhausted. Therefore, the tasks that are assigned to the S free time units of equipment m are those for which the "alternative cost" per unit time of usage of equipment m, defined in Step (2) of the algorithm, is maximal. The optimality of Algorithm TES is stated in the next theorem.

Theorem 2. *Algorithm TES produces the optimal solution for subproblem* (P5).

Proof. Note first that the solution produced by the algorithm is feasible. It is also clear that an optimal solution will necessarily use all S free time units of equipment m. Moreover, once these units are used up, the rest of the problem is of the type of problem (P3) (only with less tasks), and therefore the solution is as defined in Steps (1), (4) and (5) of Algorithm TES. It remains to prove that the choice of tasks to be assigned to equipment m, as described in Steps 2 and 3, is optimal.

Assume that the suggested solution (the solution produced by Algorithm TES) assigns to the free time units of equipment m the tasks in the set $M = \{i_1,\ldots,i_k\}$ subject to $x_{i_1} = \ldots = x_{i_{k-1}} = 1$ and $x_{i_k} = f$ where $0 < f \le 1$. Now assume by contradiction that in the optimal solution $x_{im} \ge 1/t_{im}$ for some $i \notin M$, i.e., at least one time unit of the free units of equipment m is allocated to a task which is not in M, and consider the first such unit. (We discuss here only the usage of the *free* units of equipment type m, as if they are marked; the assignment of tasks to additional equipment of that type are not relevant here). As a result, one (maybe additional) time unit of a task in M (say task i_k) has to be assigned to other equipment (instead of equipment m); as discussed earlier, the best alternative is the equipment identified in Step (1) of Algorithm TES. If we consider the contribution to the objective value of problem (P5) of the time unit whose assignment differs between the suggested solution and the optimal solution, we obtain that in the suggested solution the contribution is

$$(1/t_{im})\cdot \min_l\{t_{il}\cdot EC_l/C\},$$

and in the optimal solution the contribution is $(1/t_{i_km})\times\min_l\{t_{i_kl}\cdot EC_l/C\}$. By definition (Step (2) of Algorithm TES), the latter is higher than the former, a contradiction to the optimality of the latter solution. ■

We define:

$$LB2 = \sum_{j=1}^{r} EC_j y_j, \tag{16}$$

where y_j is obtained from Algorithm TES.

Corollary 2. *LB*2 *is a lower bound to the value of* (P4).

4. The branch and bound algorithm

Branch and bound algorithms have been extensively used for solving complex combinatorial problems, including assembly line design and balancing problems [10,12]. In this study, a frontier search branch and bound algorithm is developed for minimizing the total equipment cost. The advantage of a frontier search branch and bound algorithm is that the number of nodes investigated in the branch and bound tree, is minimal. In addition, the use of subproblems and lower bounds at each node of the branch and bound tree, which are specific to the problem investigated, considerably improve the effectiveness of the algorithm. They were developed in the previous section, and their use will be illustrated in this section.

Throughout the algorithm, workstations are opened (established) sequentially, equipment is selected and placed in the newly opened workstation, and tasks to be performed by the selected equipment are assigned to this workstation. Therefore, throughout the algorithm, partial solutions to the problem exist, which describe partial assignments of tasks to equipment and stations. In addition, for each partial solution a lower bound may be computed based on the solution of subproblem (P5), as described below. The algorithm ends when all tasks are assigned to equipment and workstations, and the obtained solution value is no larger than the lower bound of all partial solutions. In Sections 4.1–4.3 we describe the details of the algorithm:

4.1. *A node in the branch and bound tree*

Each branch and bound node X represents one partial solution of the original problem. A partial solution is characterized by a set of tasks, σ', which have already been assigned to stations, along with the equipment selected to perform these tasks, i.e., the equipment selected for these stations. Among the stations that were used thus far in the partial solution, the last opened station is the only one to which tasks may still be assigned. Finally, such a partial solution is associated with an accumulated cost, TC_X, representing the cost of purchasing the equipment decided upon thus far.

We define the slack of the last opened station at node X, S_X, as the difference between the required cycle time and the time already assigned to that station by some of the tasks in σ'. Any task i, is a *candidate* to be assigned to the last station opened if the following conditions hold:

(i) The task has no predecessors, or its predecessors are already assigned.
(ii) The time to perform task i by the already selected equipment type j (at the last opened station), t_{ij}, is no larger than the remaining slack, S_X.

If the set of candidate tasks is not empty, the station is defined as an *open station*. Otherwise, if the set of candidate tasks is empty, the station is defined as a *closed station*.

4.2. *The lower bound*

The lower bound which is calculated for each node of the branch and bound tree, consists of two elements. The first element, associated with *past* decisions, is the (exact) cost of the already selected equipment in the partial solution associated with node X, a known value which we denoted as TC_X. The second element, associated with *future* decisions, is a lower bound on the cost of the equipment which is yet to be selected for the set of yet unassigned tasks, σ (where σ is the complement of σ' in the original set of tasks). This second element is computed in one of two ways, according to whether the last opened station is closed or open:

(i) If node X represents a closed station, the remaining decisions concern the assignment of the tasks in σ to new stations that need to be opened, whose equipment types have not yet been chosen. Note that this is exactly problem (P1) (see Section 2), only limited to the set of tasks in σ. Therefore the lower bound for the element associated with future costs of node X when X is a closed station is $LB1_{(\sigma)}$, where $LB1_{(\sigma)}$ is obtained by calculating the value of (15) to the set of tasks in σ (see Section 3).
(ii) If node X represents an open station, the relaxation which is equivalent to $LB1$ but in addition takes into consideration the last opened station, is represented by a problem which is in the form of (P4). Equipment m in (P4) represents the equipment type of the last opened station in the partial solution of node X, and S in (P4) represents the remaining slack of that station, S_X. Therefore the lower bound for the element associated with future costs of node X when X is an open station is $LB2_{(\sigma)}$, where $LB2_{(\sigma)}$ is the solution of (P5) (the relaxation of (P4)), obtained by solving Algorithm TES.

Summing up the two elements discussed above of the lower bound of a given node X, we conclude that the lower bound of X is $LB_X = TC_X + LB1_{(\sigma)}$ when X is a closed station, and $LB_X = TC_X + LB2_{(\sigma)}$ when X is an open station. In both cases, the lower bound is easily calculated.

4.3. *Stages of the algorithm*

The main stages of the proposed algorithm are as follows:

Stage 1. Creation of the first level of the branch and bound tree. At this level each node contains a task which does not have precedence requirements, along with an equipment type that is capable of performing this task. Such a node is generated for every feasible equipment–task combination.

Stage 2. Selection of a node to be extended. As described above, a lower bound of the optimal cost is calculated for each node of the tree. The open node (node without descendants) with the lowest lower bound is selected for further extension, representing our choice of a frontier search algorithm.

Stage 3. Node extension. Each descendant of the extended node contains an assignment of a new single task. If the extended node represents an open station, an extension is performed for each candidate task. If the extended node represents a closed station, a new station is opened, and the extension is performed for every feasible equipment–task combination.

Stage 4. Elimination of dominated nodes. Each time a station becomes closed, a comparison between the current node and all other open nodes that are associated with closed stations, is performed in order to eliminate dominated nodes. The dominated node could be either the new one, or a previously created node. The dominance rule is described as follows: assume that at node Y, a set of tasks G has already been assigned, with an associated equipment cost, TC_Y. At another node, X, a set of tasks H has already been assigned, with an associated equipment cost TC_X. Node Y is dominated by node X if $H \supseteq G$ and $TC_X \leq TC_Y$, and therefore can be eliminated.

Stage 5. End condition. If an extended node contains all tasks, and its solution value is no larger than the lower bound of all open nodes, an optimal solution has been found. Otherwise, the algorithm proceeds as in Stage 2 above.

4.4. *Experimental study for the optimal algorithm*

We have coded our branch and bound algorithm and conducted an experimental study. As can be concluded from the running time reported in the next section, the optimal branch and bound algorithm is capable of solving moderate problem sizes in a reasonable amount of time, i.e., problems with a few dozen of tasks and with five to ten equipment types. This is only an approximation, since the variability of the run time for different instances of the same size is quite large. The purpose of the experimental study presented in this section is to examine the impact of various problem parameters on the algorithms performance, and to investigate the effectiveness of the initial lower bound ($LB1$), measured by its distance from the optimal solution value.

We report on three performance measures in this study:

(i) The size of the branch and bound tree (the total number of nodes generated).
(ii) The maximum number of open nodes in the branch and bound tree.
(iii) The running time of the algorithm.

In fact, the complexity of the algorithm (a measure of its running time) is approximately the number of nodes generated, multiplied by the complexity of the work to be done at each node. The latter is $O(\log_2 T \cdot n \cdot r)$, where T is the maximum number of open nodes in the branch and bound tree, since calculating $LB1$ or $LB2$ is $O(n \cdot r)$, and for each new node its lower bound has to be placed in a sorted list of length $O(T)$. While the running time performance measure implies the *current* capabilities of the algorithm, the other two performance measures have the advantage of being independent of the coding efficiency and the computer type. The maximal number of nodes opened simultaneously during a run is a measure of the memory space required (in addition to its impact on the complexity of the algorithm), see the next sections for more details.

We examined the impact of five parameters of the problem on the first two performance measures mentioned above. A two level full factorial experimental design has been performed, examining the significance level of each factor. The parameters and the values that were examined are described next.

(i) *The number of tasks.* The number of tasks was set to 15 and 30.
(ii) *Equipment alternatives.* The number of equipment alternatives was set to three and five.
(iii) *Variability of task duration.* The duration of every task was generated from a uniform distribution. We examined a distribution with a small variance, $U(0.8\mu, 1.2\mu)$, and a distribution with a high variance $U(0.4\mu, 1.6\mu)$, where μ is the expected value of the task duration.
(iv) *F-ratio.* The F-ratio is a measure for the flexibility in creating assembly sequences, developed by Mansoor and Yadin [13], and defined as follows: Let p_{ij} be an element of a precedence matrix $\mathbf{P}$, such that:

$$p_{ij} = \begin{cases} 1 & \text{if task } i \text{ precedes task } j, \\ 0 & \text{otherwise.} \end{cases}$$

Then, F-ratio$= 2Z/n(n-1)$, where Z is the number of zeroes in $\mathbf{P}$, and n is the number of assembly tasks. The F-ratio value is therefore between zero, when there are no precedence constraints between tasks (any sequence is feasible), and one, when only a single assembly sequence is feasible. Assembly tasks are often characterized by relatively low F-ratios. Hence, precedence diagrams with F-ratios of 0.1 and 0.4 were generated in this study.
(v) *E-ratio.* The E-ratio is a measure for the flexibility of the assembly equipment, developed by Rubinovitz and Bukchin [10], and defined as follows:

Let t_{ij} an element in a matrix $\mathbf{T}$, represent the time to perform task i by equipment type j. If task i cannot be performed by equipment type j, t_{ij} is set equal to infinity. Let M represent the number of elements set to infinity, n be the number of tasks, and r be the number of equipment types, then E-ratio $= 1 - M/n(r-1)$. The E-ratio value is therefore between zero, when each task can be performed by only a single equipment type, and one, when each task can be performed by any one of the equipment alternatives. The value of the E-ratio in this study was set to 0.3 and 0.6.

In addition to these parameter settings, we note that the cost of each equipment type was determined as a decreasing function of the value of μ. This latter choice ensures that no equipment type may dominate any other along the two dimensions of expected task duration and cost. The third desired property of an equipment type which was discussed in Section 2, namely its flexibility, was generated arbitrarily according to the specified E-ratio, in order to preserve generality of possible equipment characteristics.

The total number of experiments in a two-level, five-factors full-factorial experimental study is $2^5 = 32$. We generated 10 instances for each experiment, resulting in a total of 320 algorithm runs.

The results of the experiments are analyzed with respect to the impact that each factor has on each of the following three values of interest: (i) the total number of nodes visited; (ii) the maximal number of open nodes; and (iii) the difference between the initial lower bound and the optimal solution value. The results are presented in the form of standard ANOVA (Analysis of Variance) tables (Tables 1–3), including the values of the main effects, as well as their significance levels. Following each table, we discuss the results, and provide additional insight.

In Table 1, the values of the main effects represent the differences in the average number of nodes between the experiments with high and low value of each factor. We can see that all main effects are highly significant, with very small p-values. Even the least significant factor, the duration variability, has a p-value of less than 2%. Not surprisingly we discover that the first two main effects are positive, that is, increasing the number of tasks or the number of alternative equipment types leads to a larger branch and bound tree. The tree size is also highly and positively affected by the value of the F-ratio, which can be explained by the increase in the number of assembly alternatives (sequences) for higher F-ratio values. A similar phenomenon occurs for the E-ratio, where high values of this measure mean that there are many alternatives for the equipment assignments, leading to a larger tree. The less predictable result is the negative sign of the duration variability effect. Here we see that a smaller variability leads to a larger tree size. We believe that the reason for this is that small variability among tasks' duration increases the number of candidate tasks to be assigned at each stage of the branch and bound procedure since more tasks have similar duration.

The results with respect to the maximum number of open nodes, which is our measure for the memory space required, are summarized in Table 2. We can see that there is a similarity between the results of Tables 1 and 2, and that the main effects in both have the same signs. This implies that both the running time and the memory requirements are affected by these factors in the same way. The similarity can also be noticed when looking at the p-value column, though the p-values in Table 2 are generally higher. Four out of the five factors are highly significant, while the duration variability factor has a high p-value (0.17), and cannot be identified as significant.

Table 1. The impact of factors on the total number of nodes

Factor	*Effect*	*SS*	*df*	*MS*	*F*	*p*
(1) No. of tasks	11 240	101.07E8	1	101.07E8	50.73	7.693E-12
(2) No. of Eq. Types	7 978	509.18E7	1	509.18E7	25.56	7.433E-07
(3) Duration variability	−3 790	114.89E7	1	114.89E7	5.77	0.0169359
(4) F-ratio	11 217	100.66E8	1	100.66E8	50.52	8.434E-12
(5) E-ratio	9 019	650.76E7	1	650.76E7	32.66	2.615E-08
Error		951.69E8	314	303.09E6		
Total SS		128.09E9	319			

Table 2. The impact of factors on the maximal number of open nodes

Factor	*Effect*	*SS*	*df*	*MS*	*F*	*p*
(1) No. of tasks	1022	834.80E5	1	834.80E5	36.96	3.603E-09
(2) No. of Eq. types	859	589.94E5	1	589.94E5	26.12	5.676E-07
(3) Duration variability	−230	423.91E4	1	423.91E4	1.88	0.1716794
(4) F-ratio	1048	878.30E5	1	878.30E5	38.89	1.498E-09
(5) E-ratio	929	690.18E5	1	690.18E5	30.56	6.97E-08
Error		102.61E7	314	326.78E4		
Total SS		132.50E7	319			

Finally, we examined the impact of the main factors on the lower bound effectiveness, measured as the difference between the initial lower bound and the optimal solution value. It is interesting to discover (see Table 3) the sign of each effect and to note that all five main effects are highly significant. We observe that the gap between the lower bound and the optimal solution value is an increasing function of the number of equipment types and the duration variability; it is a decreasing function of the number of tasks, as well as the *F*-ratio and *E*-ratio. The average gap between the initial lower bound and the optimal solution value was 33.9%, which is reasonable considering the relaxation that we have made. The minimal gap was obtained for problems with 30 tasks, three equipment types, small variability of task duration, an *F*-ratio of 0.4 and an *E*-ratio of 0.6, with an average gap of 14.1% for these characteristics. The average largest gap, which was obtained for the opposite factor values, was equal to 51.7%. This information is useful for assessing the distance of a heuristic solution's value from the optimal solution's value, when a heuristic algorithm is employed. A suggested heuristic for the problem is described in the next section.

5. Heuristic algorithm – description and experiments

The frontier search branch and bound algorithm requires large computer resources in order to solve very large problems, and therefore a heuristic is required for most real world problems. In this section we present a heuristic procedure whose control parameter may be chosen according to the problem size. This control parameter determines how many nodes of the tree may be skipped, and therefore is responsible for the running time, for the memory requirements, as well as for the distance from optimality of the resulting solution.

According to the rules of the frontier search branch and bound procedure, the node with the smallest lower bound is extended at each iteration. However, some of these nodes have a very small probability of eventually providing the optimal solution, and their extension is essential only for proving the optimality of the solution. In the proposed heuristic, we modified the node selection rule, in order to avoid the extension of such nodes.

Let X be an open node at the tree level N_X, with a lower bound, LB_X. Let Y be another open node at the tree level N_Y, with a lower bound LB_Y. Define LB_0 to be the initial lower bound of the problem (before doing any assignment). Note that the levels of the tree are numbered such that the root of the tree is level 0 and the highest index level is level n, where all tasks are already assigned. The node selection rule is modified as follows:

Step 1. If $N_X \geq N_Y$, and $LB_X \leq LB_Y$, select node X.
Step 2. If $N_X \geq N_Y$, and $LB_X > LB_Y$,
Step 2.1. If

$$\frac{LB_X - LB_Y}{N_X - N_Y} \leq K \frac{LB_X - LB_0}{N_X},$$

select node X, otherwise, select node Y.

The parameter K in step 2.1 is the heuristic's control parameter; the selection of the value of K is discussed below.

Note that in Step 1 above, the usual node selection rule is applied, while in Step 2 this rule is sometimes reversed. According to Step 2, we prefer high indexed over low indexed nodes if their lower bounds are only slightly larger. The reasoning for this is that by the time the lower indexed node will become a higher indexed node, it may accumulate higher costs than the difference in their lower bounds. The left-hand side in Step 2.1 of the selection process represents the average cost per level for the levels between nodes X and Y, and this is compared with the average cost per level that was accumulated along the branch that reached the higher indexed node, X. If the former is smaller than the latter, it may be an indication that the branch that emanates from node X has better chances of providing the optimal solution. This is weighted by the control parameter, K, which represents the trade off between the tree size and the solutions quality. When $K = 0$, the inequality in Step 2.1 never holds, so that the node with the lowest lower bound is always selected, and the optimal solution is achieved. On the other hand, if K is very large, nodes with high indexed levels are always preferable over nodes with low indexed levels, and a heuristic solution is quickly obtained. However, such a solution is not likely to be a good one.

Table 3. The impact of factors on the difference between the initial lower bound and the optimal solution value

Factor	*Effect*	*SS*	*df*	*MS*	*F*	*p*
(1) No. of tasks	−0.053	0.221	1	0.221	58.57	2.52E-13
(2) No. of Eq. types	0.054	0.229	1	0.229	60.53	8.60E-14
(3) Duration variability	0.109	0.955	1	0.955	252.72	0.00E+00
(4) *F*-ratio	−0.078	0.485	1	0.485	128.18	4.33E-25
(5) *E*-ratio	−0.143	1.631	1	1.631	431.43	0.00E+00
Error		1.303	314	0.0041		
Total SS		4.824	319			

In order to choose a good value for K, namely, a value in which the problem is solvable in a reasonable time and the solution is close enough to the optimum, a sensitivity analysis on the value of K was performed.

We selected the eight problems that required the longest time to be solved by the optimal algorithm, all from the category of 30 tasks, five equipment alternatives, small variance of task duration, high F-ratio and high E-ratio. We examined those problems for 10 different values of K, between zero to 10. The results are presented in Table 4, where we can see that for each K and each of the eight problems the solution value, the size of the branch and bound tree, the maximal number of open nodes created during the algorithm run and the CPU time. The CPU time reported is in seconds, using a Pentium II 266 MHz processor. The memory requirement for each node of the branch and bound tree is approximately 150 Bytes, implying that the largest memory requirement, among the eight problems, was about 2.5MB (for Problem 4).

The results marked with an asterisk are optimal. It is apparent from the table that for each problem there is a point in which the tree size, along with the CPU time, increases dramatically, and then almost immediately the optimal solution is obtained. For all problems, the optimal solution was obtained for a relatively small number of nodes, compared with the optimal algorithm. The stochastic nature of the heuristic is also recognized, where in some cases a lower value of K caused an increase in the objective value (see for example Problem 1, $K = 1$ and $K = 0.7$). To identify the recommended value of K for this set of problems, two graphs were created and are presented in Fig. 3 (a and b). Figure 3 (a) shows the difference between the heuristic's and the optimal solution's values, where each point associated with a specific value of K is an average of the eight results. Figure 3 (b) shows the ratio between the average size of the branch and bound tree for the heuristic algorithm and the average size required by the optimal solution. We can see a clear similarity and dependency between the two graphs, which can be divided into three ranges. In the first range where K has high values, the heuristic solution value is much larger than the optimal solution value (a difference of 32% for $K = 10$), and the ratio between the two branch and bound trees is very small. When $K = 1.5$, the difference in the values becomes much smaller (8.2%) and the average size of the heuristic's branch and bound tree increases significantly. Finally, when $K = 1$, the graph becomes sharper with a higher rate of increase; at that point, a relatively good solution is obtained, with an average difference of 1.7% from the optimal solution, and with only 5.3% of the average tree size of the optimal solution. Beyond that point, when K is smaller than one, the tree size is increasing dramatically while the solution value is only slightly improving. For this set of problems, the value of $K = 1$ provides a relatively good solution where the size of the tree is relatively small. While the "best" value of K may be dependent on the parameters' characteristics, we expect that in general small values of K will provide good and fast solutions for the most difficult problems that cannot be solved to optimality.

Due to the large variability in solution time, a solution may not be obtained as fast as expected for certain problems. In these cases our recommendation is to first run the algorithm with a large value of K, in order to obtain a fast heuristic solution; then by decreasing its value gradually, we expect that the solution obtained will be improved, This process may be repeated as long as a solution is obtained in a reasonable amount of time.

6. Conclusions

In this paper we proposed a new method for the design of a flexible assembly line which may consist of several types of assembly equipment. The purpose of the design process is to choose the type of equipment to place in every station of the line and to determine the assignment of tasks to each equipment type, where the objective is minimizing total equipment cost. This design problem is NP-hard since a special case of it is the simple assembly line balancing problem, which is known to be NP-hard.

We present a formulation of the problem, based on which we develop lower bounds for both the complete and also for partial problems. These lower bounds are then used in a branch and bound algorithm. Our branch and bound algorithm also uses a dominance rule for cutting branches of the branch and bound tree, therefore reducing its running time. Although the algorithm has an exponential complexity, it is capable of solving problems of moderate size. Since it is a design problem which has to be solved only every once in a while and not frequently during operation, we are able to devote to it relatively large computational resources.

Finally, we developed a heuristic procedure which may be used for large problems that cannot be solved by the optimal algorithm. The heuristic is very flexible in determining its accuracy on one hand, and its computational time on the other hand. The trade-off between the accuracy and the computational time is controlled by the heuristics control parameter. An experimental study demonstrated the sensitivity of the accuracy and the computational time of the heuristic as a function of the control parameter, and implied on its preferred value for the examined set of problems.

We note that by solving our problem a few times, for different values of the cycle time parameter, we are able to address the higher level problem, in which the cycle time is a decision variable. In particular, an alternative to the design of a single line with a cycle time of C, is a design of m lines, with a cycle time of mC each, therefore providing

Table 4. Heuristics results

K	Problem 1				Problem 2				Problem 3				Problem 4			
	Tree size	*Tree width*	*CPU time*	*Sol. 1*	*Tree size*	*Tree width*	*CPU time*	*Sol. 2*	*Tree size*	*Tree width*	*CPU time*	*Sol. 3*	*Tree size*	*Tree width*	*CPU time*	*Sol. 4*
10	107	75	0.05	1900	122	84	0.05	1740	111	78	0.05	1570	88	53	0.05	1440
5	107	70	0.05	1780	127	84	0.05	1440	115	73	0.05	1550	98	58	0.05	1420
3	112	72	0.05	1750	127	63	0.05	1490	137	77	0.05	1420	109	51	0.05	1370
2	112	69	0.05	1640	165	64	0.05	1360	158	78	0.05	1420	93	39	0.05	1500
1.5	106	63	0.05	1620	240	67	0.05	1370	563	119	0.17	1320	146	61	0.05	1400
1	379	80	0.11	1280*	6 690	397	2.25	1330	1 624	198	0.44	1290	6 663	612	2.58	1290*
0.7	12 785	852	6.10	1290	18 243	920	8.35	1330	10 349	457	3.95	1270*	31 338	1 538	23.78	1290*
0.5	19 359	1 360	13.24	1280*	33 375	1637	20.43	1320*	13 457	662	6.27	1270*	86 927	4 263	122.87	1300
0.3	49 776	3 799	51.41	1280*	42 029	1930	26.75	1320*	19 948	1148	9.56	1270*	75 295	5 558	102.93	1290*
0	98 699	11 365	196.09	1280*	55 109	2933	39.54	1320*	25 965	1905	14.06	1270*	163 618	16 788	550.85	1290*

Table 4. (Cont.)

K	Problem 5				Problem 6				Problem 7				Problem 8			
	Tree size	*Tree width*	*CPU time*	*Sol. 5*	*Tree size*	*Tree width*	*CPU time*	*Sol. 6*	*Tree size*	*Tree width*	*CPU time*	*Sol. 7*	*Tree size*	*Tree width*	*CPU time*	*Sol. 8*
10	85	49	0.05	1880	100	63	0.05	1580	75	42	0.05	1840	105	70	0.05	1900
5	93	52	0.05	1700	105	64	0.05	1400	75	42	0.05	1840	110	70	0.05	1920
3	101	52	0.05	1670	105	64	0.05	1400	86	40	0.05	1630	109	51	0.05	1690
2	162	66	0.05	1520	124	61	0.05	1400	90	40	0.05	1640	132	60	0.05	1490
1.5	167	68	0.05	1520	221	82	0.05	1400	114	40	0.05	1410	460	92	0.11	1320*
1	6 653	233	1.81	1420	4 756	501	1.71	1340	1 588	77	0.38	1410	3 732	330	1.16	1320*
0.7	23 782	714	11.26	1370*	14 409	1499	7.74	1300*	10 852	651	4.55	1350*	61 242	1 809	54.37	1320*
0.5	68 474	1830	55.42	1370*	22 654	2415	16.64	1320	39 705	1995	32.41	1350*	47 148	3 088	39.28	1320*
0.3	70 684	3372	59.81	1370*	35 454	3432	29.72	1300*	60 834	3229	56.85	1350*	67 928	5 955	84.86	1320*
0	103 208	5847	128.86	1370*	53 617	6408	61.19	1300*	69 131	5595	75.35	1350*	161 320	16 414	581.39	1320*

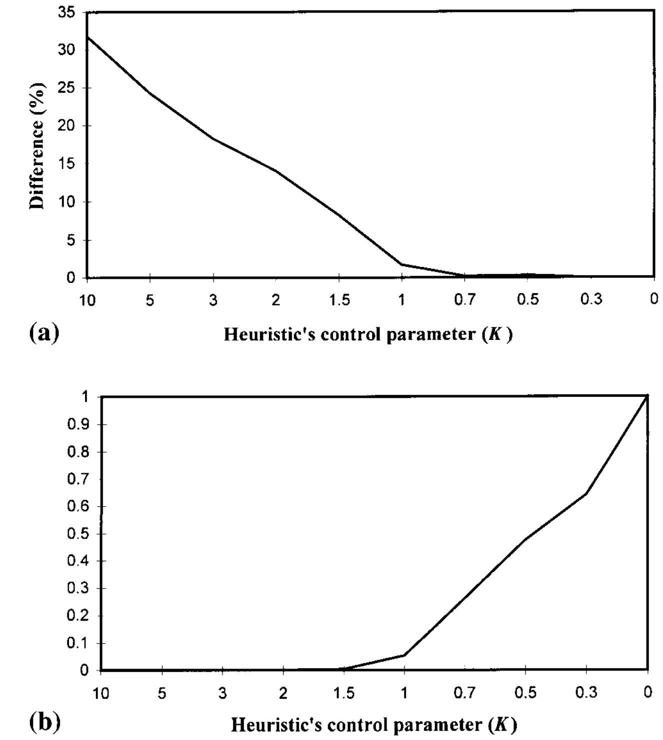

Fig. 3. (a) A comparison between the heuristic and the optimal solution, and (b) the ratio for the tree size of the heuristic and that of the optimal solution.

the same throughput. Since m, the number of separate lines, is not likely to be high, it is still reasonable to solve the problem for every resulting value of cycle time.

References

[1] Baybars, I. (1986) A survey of exact algorithm for the simple assembly line balancing problem. *Management Science*, **32**, 909–932.

[2] Ghosh, S. and Gagnon, R.J. (1989) A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems. *International Journal of Production Research*, **27**, 637–670.

[3] Scholl, A. (1999) *Balancing and Sequencing of Assembly Lines*, 2nd edition. Physica-Verlag, Heidelberg, New York.

[4] Sarin, S.C. and Erel, E. (1990) Development of cost model for the single-model stochastic assembly line balancing problem. *International Journal of Production Research*, **28**, 1305–1316.

[5] Karp, R.M. (1972) Reducibility among combinatorial problems, in *Complexity of Computer Computation*, Miller, R.E. and Thatcher, J.W. (eds.), Plenum Press, New York, pp. 85–103.

[6] Graves, S.C. and Holmes Redfield, C. (1988) Equipment selection and task assignment for multiproduct assembly system design. *The International Journal of Flexible Manufacturing Systems*, **1**, 31–50.

[7] Graves, S.C. and Whitney, D.E. (1979) A mathematical programming procedure for the equipment selection and system evaluation in programmable assembly, in *Proceedings of the Eighteenth IEEE Conference on Decision and Control*, Ft Lauderdale, FL. pp. 531–536.

[8] Graves, S.C. and Lamar, B.W. (1983) An integer programming procedure for assembly design problems. *Operations Research*, **31**(3), 522–545.

[9] Pinto, P.A., Dannenbring, D.G. and Khumawala, B.M. (1983) Assembly line balancing with processing alternatives: an application. *Management Science*, **29**, 817–830.

[10] Rubinovitz, J. and Bukchin, J. (1993) RALB – a heuristic algorithm for design and balancing of robotic assembly lines. *Annals of the CIRP*, **42**, 497–500.

[11] Tsai, D.M. and Yao, M.J. (1993) A line-balanced-base capacity planning procedure for series-type robotic assembly line. *International Journal of Production Research*, **31**, 1901–1920.

[12] Johnson, J.R. (1988) Optimally balancing large assembly lines with FABLE. *Management Science*, **34**, 240.

[13] Mansoor, E.M. and Yadin, M. (1971) On the problem of assembly line balancing, in *Developments in Operations Research*, Avi-Itzhak, B. (ed), Gordon and Breach, New York, p. 361.

Appendix – Summary of notation

Parameters

t_{ij} = duration of task i when performed by equipment j $(i = 1, \ldots, n,\ j = 1, \ldots, r)$;

EC_j = cost of equipment type j $(j = 1, \ldots, r)$;

C = required cycle time;

P_i = set of immediate predecessors of task i $(i = 1, \ldots, n)$.

Decision variables

$$y_{jk} = \begin{cases} 1 & \text{if equipment } j \text{ is assigned to station } k; \\ 0 & \text{otherwise.} \end{cases}$$

$$x_{ijk} = \begin{cases} 1 & \text{if task } i \text{ is performed by equipment } j \text{ at station } k; \\ 0 & \text{otherwise.} \end{cases}$$

$$y_j = \textstyle\sum_{k=1}^{n} y_{jk} = \text{total number of type } j \text{ equipment};$$

$$x_{ij} = \textstyle\sum_{k=1}^{n} x_{ijk} = \begin{cases} 1 & \text{if task } i \text{ is performed by equipment } j; \\ 0 & \text{otherwise.} \end{cases}$$

Lower bounds

$LB1$ = the solution of problem (P3) and a lower bound for problem (P1);

$LB2$ = the solution of problem (P5) and a lower bound for problem (P4).

Optimal branch and bound related values

X = a node n the branch and bound tree;

TC_X = the cost of purchasing the equipment decided upon thus far in the partial solution associated with node X;

S_X = slack of the last opened station at node X;

σ' = a set of tasks which have already been assigned to stations in the partial solution associated with node X;

σ = the complement of σ' in the original set of tasks, i.e., the set of tasks which have not been assigned yet to stations in the partial solution associated with node X;

$LB1_{(\sigma)}$ = the value of $LB1$ when only the set of tasks σ is considered;
$LB2_{(\sigma)}$ = the value of $LB2$, given the set of tasks σ;
Z_X = the lower bound of node X.

Parameters used in the experimental study

μ = the expectation of the task duration;
F-ratio = $2Z/n(n-1)$, where Z is the number of zeroes in the matrix **P** whose elements are:

$$p_{ij} = \begin{cases} 1 & \text{if task } i \text{ precedes task } j, \\ 0 & \text{otherwise.} \end{cases}$$

E-ratio = $1 - M/n(r-1)$ where M = the number of t_{ij} elements that equal to infinity.

Heuristic branch and bound related values

N_X = tree level of node X;
LB_X = lower bound of node X in the branch and bound tree;
LB_0 = the initial lower bound of the problem;
K = the heuristic's control parameter.

Biographies

Joseph Bukchin is a member of the Department of Industrial Engineering at Tel Aviv University. He received B.Sc., M.Sc. and D.Sc. degrees in Industrial Engineering at the Technion, Haifa, Israel. His main research interests are in the areas of assembly systems design, assembly line balancing, facility design, design of cellular manufacturing systems, operational scheduling as well as work station design with respect to cognitive and physical aspects of the human operator.

Michal Tzur received her B.A. from Tel Aviv University, Israel, and her M.Phil and Ph.D. in Management Science from Columbia University. Michal joined the Faculty of Industrial Engineering at Tel Aviv University, Israel, in 1994 after spending 3 years at the Operations and Information Management department at the Wharton School at the University of Pennsylvania. Her research interests are in the areas of logistic systems, inventory management combined with forecast horizon results, operations scheduling and production planning.

Contributed by the Manufacturing Systems Control Department