International Journal of Production Research,
Vol. 43, No. 8, 15 April 2005, 1667–1684

Taylor & Francis
Taylor & Francis Group

Integer programming approach to the printed circuit board grouping problem

S. YU†, D. KIM‡ and S. PARK‡*

†School of Business Administration, Catholic University of Pusan, Bugok3-dong, Kumjung-gu, Busan 609-757, Republic of Korea
‡Department of Industrial Engineering, KAIST, Guseong-dong, Yuseong-gu, Daejeon 305-701, Republic of Korea

(*Received December 2003*)

A printed circuit board (PCB) grouping problem arising from the electronics industry is considered. Given a surface-mounting device with a number of component feeders and several types of PCBs to be produced, the problem is how to group the PCBs so that the total set-up time for component feeders is minimized. The problem is formulated as an integer-programming problem and a column generation approach is proposed to solve it. In this approach, the original problem is decomposed into a master problem and a column-generation subproblem. Starting with a few columns in the master problem, new columns are generated successively by solving the subproblem optimally. To solve the subproblem, a branch-and-cut approach is used. To solve the master problem, a branch-and-bound algorithm is used with the generated columns. However, a branching strategy is also proposed that maintains consistency in the column-generation procedure after branching. Computational experiments show that the solution approach gives high-quality solutions in reasonable computing time.

Keywords: Printed circuit board grouping; Zero-one programming; Column generation

1. Introduction

The electronics industry relies heavily on surface-mounting devices (SMDs) for the mounting of electronic components on the surface of printed circuit boards (PCBs) (Crama *et al.* 1990). The production rate of the PCBs depends on two factors: the time needed to assemble a PCB (Crama *et al.* 1990, Yu *et al.* 1997) and the set-up time of component feeders when several types of PCBs are produced (Carmon *et al.* 1989, Bhaskar and Narendran 1996, Daskin *et al.* 1997, Rajkumar *et al.* 1998). The present paper considers an optimization problem for the second case, which will be called the 'PCB grouping problem'.

Before describing the problem, we give a brief explanation of a typical PCB assembly process using SMD. The device usually consists of an X–Y robot with

*Corresponding author. E-mail: sspark@kaist.ac.kr

International Journal of Production Research
ISSN 0020–7543 print/ISSN 1366–588X online
http://www.tandf.co.uk/journals
DOI: 10.1080/00207540412331299639

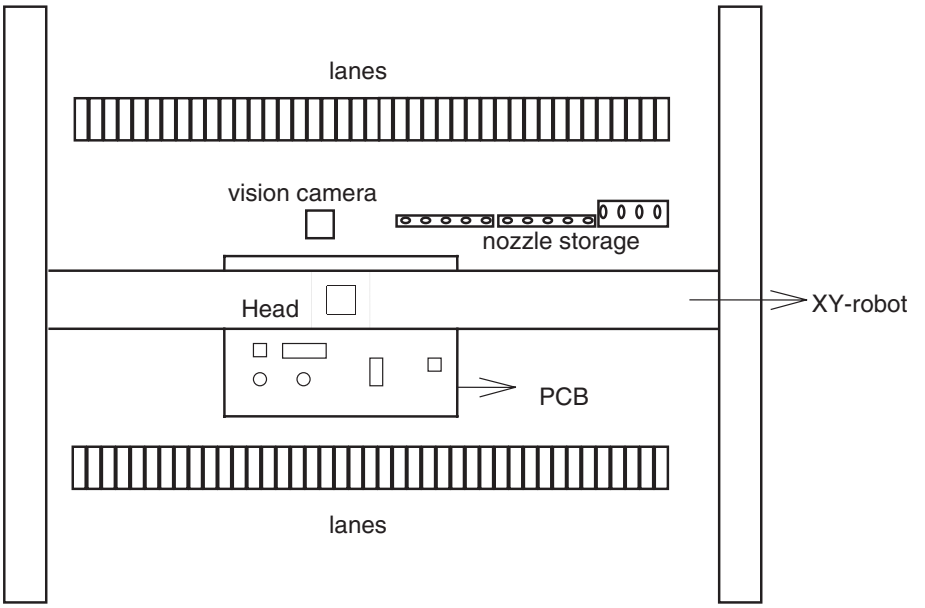

Figure 1. Conceptual diagram of the surface-mounting device.

a head(s), a table on which a PCB is located and a number of lanes in which component feeders are located. The positions of the table and feeders are fixed. When a component is mounted on the PCB, the head picks a component from the feeder that supplies it, moves to the mounting location on the PCB and mounts it. This operation is repeated until all components are mounted on the PCB. Figure 1 shows an example of the SMD.

In the PCB assembly process, a typical assembly machine has the capability to mount a few thousand components per hour, while its set-up may take about an hour (Shtub and Maimon 1992). Moreover, it is estimated that the set-up activity takes about 40% of the usable time of the machine (Randhawa *et al.* 1985). If all the required components for assembling a PCB are not available on the feeders, a new set-up of components will be required. This may cause frequent hold ups in production. Thus, any reduction in set-up time will increase the machine use significantly and reduce the manufacturing lead time.

The problem considered here can be described as follows. There is a set of several types of PCBs to be produced on an SMD. The SMD has a limited number of lanes in which feeders are placed. The PCBs need to be divided into a few numbers of groups. For each group, the component feeders necessary to produce all PCBs in the group must be loaded into the machine simultaneously. After mounting the components for a PCB group, the component feeders are unloaded and the feeders necessary to produce the next group of PCBs are loaded into the machine. Then the production of the next group of PCBs begins. This process is repeated until all the PCBs are produced. Here, we are not concerned about the production volume of each type of PCB, only the types of component feeders needed to produce each type of PCB.

In real production, a PCB may be loaded into the SMD more than once to complete the mounting of all components. However, we focus on a restricted class

of problems in which each PCB is loaded exactly once. In that case, all the component feeders required to produce a group of PCBs must be loaded on the machine before production begins. Thus, the problem can be defined as the one of determining the PCB groups with the objective of minimizing the total set-up cost of component feeders. Note that the total set-up cost (time) is the sum of set-up cost (time) of each component feeder.

Some details of the problem including assumptions are given as follows:

- Each type of PCB should be included in exactly one group.
- Set-up of a component feeder consists of loading and unloading into the machine.
- Mechanical set-ups (i.e. changing the dimension of the machine's table or changing the width of the conveyor carrying the PCBs to the machine) while producing PCBs in the same production group are not required. If a group includes different sized PCBs, they are produced by 'panelling', a method in which several PCBs are assembled as a single standard sized panel that is later cut to the correct dimensions.
- PCB transfer times into and out of the machine are negligible.
- Refilling components in the machines during assembly is not considered, since the quantity of each component required is independent of the scheduling method used.
- Number of lanes required by all feeders is larger than the machine capacity.

The PCB grouping problem is not new in the sense that many researchers have studied similar or the same problems. Previous studies have focused on the time-consuming set-up operation to reduce cost and increase the throughput of small-lot PCB assembly. For example, Carmon *et al.* (1989) introduced group set-up (GSU) method to reduce set-up time in the assembly of PCBs. They proposed an algorithm based on the generalized group technology (GT) concept (Kusiak 1987). In a subsequent paper (Maimon *et al.* 1993), they compared two scheduling methods: GSU and sequence dependent set-up (SDS).

Hashiba and Chang (1991) and Maimon and Shtub (1991) proposed other approaches based on group technology. Hashiba and Chang decomposed the problem into three parts. They then presented separate heuristics for each subproblem. Maimon and Shtub combined SDS and GSU methods and gave a mixed-integer non-linear programming model for the problem.

Shtub and Maimon (1992) examined the use of similarity measures for grouping PCBs. Using Jaccard's similarity coefficients (Sokal and Sneath 1963), they provided a general framework for developing a construction type heuristic. Luzzato and Perona (1993) also proposed a heuristic based on the group technology concept to assign PCBs to a number of machines so that set-up times were minimized and workloads balanced.

Contrary to the previous research based on heuristic algorithms, we try to solve the integer-programming formulation of the problem by the column generation approach. Our approach decomposes the original problem into a master problem and a column-generation subproblem. Starting with a few columns in the master problem, we generate new columns successively by solving the subproblem optimally. The process of adding columns is repeated until no more profitable column can be found. Since Gilmore and Gomory (1961) used this approach to solve the cutting-stock problem, it has been used for various optimization problems (Ryan

and Foster 1981, Johnson *et al*. 1993, Vance *et al*. 1994). To solve the subproblem optimally, we use the branch-and-cut approach, which is a generalization of the branch-and-bound method using linear programming (LP) relaxations. For general expositions on the procedure, see Nemhauser and Wolsey (1988).

The paper is organized as follows. Section 2 formulates the PCB grouping problem as two integer LP problems. Section 3 proposes a branch-and-cut algorithm to solve the column generation problem. Section 4 provides the overall solution approach to solve the PCB grouping problem. Section 5 summarizes our computational experiences with the proposed algorithm for some real-world problems and some randomly generated problems. Finally, concluding remarks are given in section 6.

2. Mathematical formulations

This section presents two formulations for the PCB grouping problem. We first introduce a standard formulation of the PCB grouping problem that can be directly obtained from the definition of the problem. Then, we propose a disaggregated formulation with the variables denoting the grouping of PCBs to overcome some pitfalls of the first formulation.

2.1 *Integer-programming formulation*

To formulate the PCB grouping problem, the following notation is used:

Notation

F set of component feeders,
J set of PCB types,
K set of possible PCB groups,
α_f set-up cost for feeder f, $f \in F$,
β_j set-up cost for PCB type j, $j \in J$,
B machine capacity (number of lanes in the machine),
S_f number of lanes occupied by feeder f, $f \in F$,
N_j set of feeders required by PCB type j, $j \in J$.

In the notation, β_j is the set-up cost associated with producing a batch of PCBs of type j. This cost will be incurred for the group in which PCB j is included. Similarly, the feeder cost α_f will be incurred for each group in which feeder f is included. It is assumed that the costs α_f and β_j are strictly positive.

The decision variables are defined as follows.

$$X_{fk} = \begin{cases} 1, & \text{if feeder } f \text{ is in group } k \\ 0, & \text{otherwise} \end{cases}$$

$$Y_{jk} = \begin{cases} 1, & \text{if PCB } j \text{ is in group } k \\ 0, & \text{otherwise} \end{cases}$$

$$Z_{fjk} = \begin{cases} 1, & \text{if feeder } f \text{ required by PCB } j \text{ is in group } k \\ 0, & \text{otherwise} \end{cases}$$

Note that if feeder f is in the group k, it can be used in the production of any PCB that needs feeder f. Now, an integer-programming formulation for the PCB grouping problem is given as follows:

$$\textbf{(P)} \quad \min \sum_{f \in F} \sum_{k \in K} \alpha_f X_{fk} + \sum_{j \in J} \sum_{k \in K} \beta_j Y_{jk} \tag{1}$$

$$\text{s.t.} \quad \sum_{k \in K} Z_{fjk} = 1, \quad \text{for all } f \in N_j,\ j \in J \tag{2}$$

$$X_{fk} \geq Z_{fjk}, \quad \text{for all } f \in N_j,\ j \in J,\ k \in K \tag{3}$$

$$Y_{jk} \geq Z_{fjk}, \quad \text{for all } f \in N_j,\ j \in J,\ k \in K \tag{4}$$

$$\sum_{f \in F} s_f X_{fk} \leq B, \quad \text{for all } k \in K \tag{5}$$

$$\sum_{k \in K} Y_{jk} = 1, \quad \text{for all } j \in J \tag{6}$$

$$X_{fk} \in \{0, 1\}, \quad \text{for all } f \in F,\ k \in K \tag{7}$$

$$Y_{jk} \in \{0, 1\}, \quad \text{for all } j \in J,\ k \in K \tag{8}$$

$$Z_{fjk} \in \{0, 1\}, \quad \text{for all } f \in N_j,\ j \in J,\ k \in K \tag{9}$$

The objective function (1) means the sum of the set-up costs for feeders and PCBs. Note that the set-up cost for a feeder may be incurred more than once if the feeder is needed in more than one group. Constraints (2) state that each feeder-PCB combination is assigned to exactly one group. A feeder may be assigned to more than one group. Constraints (3) and (4) state that if feeder f required by PCB j is included in group k, then feeder f and PCB j must be assigned to group k, respectively. Constraints (5) are the capacity constraints for the lanes occupied by feeders for each group. Constraints (6) state that each PCB must be included in exactly one group. Constraints (7–9) are the integrality constraints on the decision variables.

We give a few remarks on formulation (P). We need to define Z_{fjk} variables as above if we do not have constraints (6). For example, Daskin *et al.* (1997) proposed a similar formulation as above using Z_{fjk} variables. But, if we include constraints (6) as in formulation (P), the variables Z_{fjk} are not necessary any more because each PCB must be contained in exactly one group and for group k which contains PCB j, feeders required by PCB j must be included in group k. In this case, we may represent the situation as $X_{fk} \geq Y_{jk}$ for all $f \in N_j$ and $j \in J$. In addition, in the objective function (1), the term $\Sigma_{j \in J} \Sigma_{k \in K} \beta_j Y_{jk}$ can be removed because $\Sigma_{j \in J} \Sigma_{k \in K} \beta_j Y_{jk}$ has always the same value regardless of any PCB grouping. Thus, (P) can be simplified as follows:

$$\textbf{(P1)} \quad \min \sum_{f \in F} \sum_{k \in K} \alpha_f X_{fk}$$

$$\text{s.t.} \quad X_{fk} \geq Y_{jk}, \quad \text{for all } f \in N_j,\ j \in J,\ k \in K$$

and (5−8)

(P1) is a formulation that explicitly represents the PCB grouping problem. Note that the LP relaxation of (P1) can be solved easily. However, (P1) has symmetric structure. Given a feasible solution to (P1), we can obtain a different solution with the same objective value by assigning a different group number to the sets of PCBs and feeders contained in the same group. It implies that there exist $|K|!$ different solutions to (P1), which are actually the same solution. This symmetric structure causes branch-and-bound procedure to perform poorly because the problem barely changes after branching. In addition to the symmetric structure of the formulation, the bound on optimal objective value of (P1) provided by the LP relaxation of (P1) can be very weak. Thus, we present an alternative formulation to solve the PCB grouping problem efficiently.

2.2 *Alternative formulation using the grouping variables*

We present an alternative formulation for the PCB grouping problem to overcome pitfalls of the formulation (P1). The basic idea is to decompose the PCB grouping problem into a master problem and a subproblem. The master problem is constructed by a set of columns that indicate possible grouping configurations and the subproblem regards how to find columns (when necessary) in the master problem.

First, to formulate the master problem, we define G as the set of all possible PCB grouping configurations and $W(g)$ as the set of indices of feeders required by a PCB group configuration $g \in G$. In addition, we let $p_{jg} = 1$ if PCB j is included in grouping configuration g, 0 otherwise. Note that a grouping $g \in G$ must satisfy the inequality $\Sigma_{f \in W(g)} s_f \leq B$. We define decision variables as follows:

$$\lambda_g = \begin{cases} 1, & \text{if grouping configuration } g \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$

Now, the PCB grouping problem can be formulated as follows:

$$(\mathbf{MP}) \quad \min \sum_{g \in G} c_g \lambda_g \tag{10}$$

$$\text{s.t.} \quad \sum_{g \in G} p_{jg} \lambda_g = 1, \quad \text{for all } j \in J, \tag{11}$$

$$\lambda_g \in \{0, 1\}, \quad \text{for all } g \in G, \tag{12}$$

where $c_g = \sum_{f \in W(g)} \alpha_f$.

The objective function (10) means the sum of costs of PCB-grouping configurations selected. Constraints (11) ensure that each PCB is included in exactly one group. In the objective function (10), c_g is the sum of set-up costs of feeders necessary to assemble PCBs included in configuration g.

The problem (MP) has no symmetric structure. But it has exponentially many columns as the number of PCBs and feeders becomes large. However, we can solve the LP relaxation of (MP) efficiently by using the column generation technique. Let (RMP) be the LP relaxation of (MP), i.e. the problem obtained after dropping the integrality requirements (12) on λ_g. Now, it is assumed that a subset $G' \subset G$ of grouping configuration is given. We obtain the restricted linear programme (RMP′) by replacing G with G' in (RMP). The optimal solution to (RMP′)

is suboptimal to (RMP). Let π_j be the dual variable associated with j-th constraint in (11). Because $c_g = \Sigma_{f \in W(g)} \alpha_f$, the reduced cost of column g, denoted by $\bar{c}_g$, is:

$$\bar{c}_g = \sum_{f \in W(g)} \alpha_f - \sum_{j \in J} p_{jg} \pi_j, \quad \text{for all } g \in G.$$

Let π^* be an optimal dual solution returned by the simplex method after we solve (RMP′). Then, the current optimal solution to (RMP′) is also optimal to (RMP) if $\Sigma_{f \in W(g)} \alpha_f - \Sigma_{j \in J} p_{jg} \pi_j^* \geq 0$, for all $g \in G \backslash G'$. Therefore, the optimality condition for (RMP) is:

$$\max_{g \in G \backslash G'} \left\{ \sum_{j \in J} p_{jg} \pi_j^* - \sum_{f \in W(g)} \alpha_f \right\} \leq 0 \tag{13}$$

By condition (13), if we can find a grouping configuration $g \in G \backslash G'$ such that $\Sigma_{j \in J} p_{jg} \pi_j^* - \Sigma_{f \in W(g)} \alpha_f > 0$, the corresponding variable λ_g can be an entering non-basic variable for the current basis. Therefore, the corresponding column can be added to (RMP′) and (RMP′) is reoptimized. If we can verify that $\Sigma_{j \in J} p_{jg} \pi_j^* - \Sigma_{f \in W(g)} \alpha_f \leq 0$ for all $g \in G \backslash G'$, the current optimal solution to (RMP′) is also optimal to (RMP). Therefore, we can solve (RMP) to optimality by generating columns that violate the optimality condition (13) continuously until no more columns are generated. We can start the procedure only with a few initial columns.

Now, we introduce the formulation of the column generation problem to generate a column to enter the basis. To formulate the column generation problem, we introduce variables x_f and y_j, which indicate whether or not feeder f and PCB j are included in the group, respectively. If PCB j is included in the group ($y_j = 1$), then corresponding feeders should be included in the group ($x_f = 1$ for $f \in N_j$). In addition, the number of lanes occupied by all feeders included in the group should not exceed the capacity. Then, the column generation problem can be formulated as follows:

$$\textbf{(SP)} \quad \max \sum_{j \in J} \pi_j^* y_j - \sum_{f \in F} \alpha_f x_f \tag{14}$$

$$\text{s.t.} \quad y_j \leq x_f, \quad \text{for all } f \in N_j \quad \text{and} \quad j \in J \tag{15}$$

$$\sum_{f \in F} s_f x_f \leq B \tag{16}$$

$$y_j \in \{0, 1\}, \quad \text{for all } j \in J$$

$$x_f \in \{0, 1\}, \quad \text{for all } f \in F$$

Constraints (15) imply that if PCB j is included in the group, all feeders needed to assemble the PCB must also be included in the group. Constraint (16) is related to the machine capacity.

Note that (SP) is the precedence constrained knapsack problem (Boyd 1993, Park and Park 1997). Generally, the problem is known to be NP-hard (Boyd 1993). (SP) is also NP-hard even though this problem has a special structure (Park and Park 1997). In the following, we present solution procedures to solve (SP).

3. Subproblem optimization

To solve (SP) optimally, we propose a branch-and-cut procedure similar to the branch-and-bound procedure except that we solve the problem at each node of

the enumeration tree by using the strong cutting planes. These strong cutting planes may reduce the number of nodes to be examined in the branch-and-bound tree, and consequently may save the computational time to solve (SP) (Padberg and Rinaldi 1992).

3.1 *Valid inequalities*

The problem (SP) is a generalization of the knapsack problem obtained by imposing a partial order on the items, which are associated with feeders and PCBs. It is said to be a precedence constraint from item f to item j if item j can be included in the knapsack only if item f is included. That is, if $y_j = 1$, then $x_f = 1$ for $f \in N_j$.

For a given instance of (SP), we can define the associated precedence graph $D = (V, A)$, where the set V of nodes is the union of the set F and the set J, i.e. $V = F \cup J$, and the set A of arcs represents the precedence relationship between nodes in V. Note that $(f, j) \in A$ if and only if $f \in N_j$. For $(f, j) \in A$, node f is called the *predecessor* of node j and node j is called the *successor* of node f. Note that the graph $D = (V, A)$ is a directed bipartite graph.

A pair of nodes $k_1 \in V$ and $k_2 \in V$ is called *incomparable* if both $(k_1, k_2) \notin A$ and $(k_2, k_1) \notin A$. A set $C \subseteq V$ is called incomparable if the elements in C are pairwise incomparable. Note that the sets F and J, both are incomparable. Figure 2 gives an example of a precedence graph.

Suppose that an instance of (SP) and the associated graph D are given. From the well-known results on the polyhedral structure of the knapsack polytope, a subset $C \subseteq F$ is called a cover if $\Sigma_{f \in C} s_f > B$ (Nemhauser and Wolsey 1988). And the associated inequality $\Sigma_{f \in C} x_f \leq |C| - 1$ is called a cover inequality. A cover is called a minimal cover if no proper subset of it is a cover. If C is a minimal cover, then the cover inequality $\Sigma_{f \in C} x_f \leq |C| - 1$ is valid for the knapsack polytope. However, when there exists precedence constraints between variables as our application, the following modification is more useful. We use the following notation throughout the remainder of this paper:

$$
\begin{aligned}
N(C) &= \bigcup_{j \in C} N_j, \quad C \subseteq J \\
T(C) &= C \cup N(C), \quad C \subseteq J \\
S_f(C) &= \{j \in C\colon (f, j) \in A\}, \quad f \in F,\ C \subseteq J \\
N_2(C) &= \{f \in N(C)\colon |\, S_f(C) \,| \geq 2\}, \quad C \subseteq J
\end{aligned}
$$

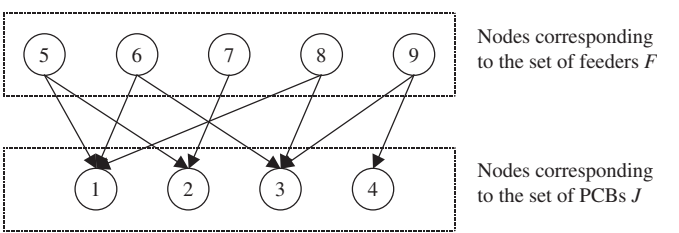

Figure 2. Example of the precedence graph for the printed circuit board grouping problem.

Definition 1: $C \subseteq J \subset V$ is a minimal induced cover (MIC) if:

- C is incomparable.
- $\Sigma_{f \in N(C)} s_f > B$.
- $\Sigma_{f \in N(C \setminus \{j\})} s_f \leq B$, for all $j \in C$.

In words, a minimal induced cover is a set of incomparable items (PCBs) with the property that the set of feeders needed to produce the PCBs does not satisfy the machine capacity constraint, while the needed feeders for any proper subset of the cover satisfy the capacity constraint. Definition 1 follows the work of Park and Park (1997).

By a direct consequence of Definition 1, it can be easily shown that for a MIC $C \subseteq J$, the following inequality is valid for the convex hull of feasible solutions to (SP). We refer to this inequality as the MIC inequality:

$$\sum_{j \in C} y_j \leq |C| - 1$$

With the MIC inequality, we can get stronger cutting planes by lifting the variables in F and $J \backslash C$. Lifting is a systematic procedure to obtain valid inequalities for a polyhedron from valid inequalities for lower dimensional polyhedron. For details of the general lifting procedure, see Nemhauser and Wolsey (1988).

Assume that a MIC $C \subseteq J$ is given. In general, lifting is dependent on the order of the variables lifted. To lift a MIC inequality, we first lift the variables in $N_2(C) \subseteq N(C)$, second those in $F \setminus N_2(C)$, and finally those in $J \setminus C$. The rationale for the order of lifting, individual lifting procedures and theoretical background appear in Park and Park (1997).

3.2 *Separation*

Now, we give a formulation of the separation problem to find a violated minimal induced cover and a heuristic algorithm to solve the problem. Given a fractional solution (x^*, y^*) to the LP relaxation of (SP), the problem is to find a MIC inequality violated by (x^*, y^*), i.e. we want to find a MIC C that satisfies:

$$\sum_{j \in C} y_j^* > |C| - 1 \tag{17}$$

Inequality (17) is equivalent to $\Sigma_{j \in C}(1 - y_j^*) < 1$. All values of S_f for $f \in F$ are positive integers. Thus, the separation problem can be formulated as follows:

$$\begin{aligned}
(\mathbf{SEP}) \quad & \zeta = \min \sum_{j \in J} (1 - y_j^*) z_j \\
\text{s.t.} \quad & \sum_{f \in F} s_f x_f \geq B + 1 \\
& x_f \leq \sum_{\{j : f \in N_j\}} z_j, \quad \text{for all } f \in F \\
& x_f \in \{0, 1\}, \quad \text{for all } f \in F \\
& z_j \in \{0, 1\}, \quad \text{for all } j \in J
\end{aligned}$$

Suppose $\zeta \geq 1$, then the proposed solution satisfies all of the MIC inequalities. Otherwise, the optimal solution to the above problem provides a most violated MIC inequality.

When $|F| = B + 1$ and $s_f = 1$ for all $f \in F$, (SEP) is restricted to the set covering problem. Thus, (SEP) is NP-hard.

Instead of solving (SEP) to optimality, we use a heuristic to find a violated MIC inequality. The basic idea of the heuristic algorithm is to select an element $j \in J$, having large value of y_j^* and many predecessors in $D = (V, A)$, then it is added to C. Then the procedure is repeated, after updating $D = (V, A)$, until the used capacity for feeders exceeds B. Given an associated graph $D = (V, A)$, the heuristic algorithm is described as follows.

3.2.1 HSEP: separation heuristic for finding a MIC

Phase 1: *Find an induced cover*:

Step 0: Initialize:
Set $D^0 = (V^0, A^0)$ where $V^0 = (F^0, J^0) = V$ and $A^0 = A$.
Set $N_j^0 = N_j$ for all $j \in J$.
Set $C = \phi$ and $t = 0$.

Step 1: Find a node with the largest weight:
From the graph $D^t = (V^t, A^t)$, compute the weight $u_j^t = y_j^* \sum_{f \in N_j^t} s_f$ for $j \in J^t$.
Set $j^* = \arg\max_{j \in J^t} \{u_j^t\}$ (if ties occur, select arbitrarily).

Step 2: Update the set C:
Add j^* to C, i.e. $C \leftarrow C \cup \{j^*\}$.

Step 3: Check feasibility:
If $\Sigma_{\{f \,|\, f \in N_j^t, j \in C\}} s_f > B$, go to phase 2. Otherwise, go to Step 4.

Step 4: Update graph:
Set $t \leftarrow t + 1$.
$F^t = F^{t-1} \backslash \{f \in F^{t-1} \,|\, f \in N_{j^*}^{t-1}\}$,
$J^t = J^{t-1} \backslash \{j^*\}$.
$V^t = (F^t, J^t)$.
$A^t = A^{t-1} \backslash \{(f, j) \in A^{t-1} \,|\, f \in N_{j^*}^{t-1}\}$
$D^t = (V^t, A^t)$.
$N_j^t = \{f \in F^t \,|\, (f, j) \in A^t\}$.
Go to Step 1.

Phase 2: *Select minimal elements in C*:
Without loss of generality, let the set C be $\{1, 2, \ldots, |C|\}$.
For $k = 1$ to $|C|$, do
If $\sum_{f \in C \backslash \{k\}} s_f > B$, set $C \leftarrow C \backslash \{k\}$

3.3 *Branch-and-cut procedure for the subproblem (SP)*

Now, we present an overall description of the branch-and-cut procedure to solve (SP). Let (RSP) be the LP relaxation of (SP), i.e. the problem obtained after

dropping the integrality requirements on y_j and x_f in (SP). The branch-and-cut procedure begins by solving (RSP). If the optimal solution to (RSP) is integral, we get an optimal solution to (SP) and the procedure is terminated. If not, an upper bound on the optimal value of (SP) is provided by this solution. Then we proceed to find a minimal induced cover by the separation heuristic HSEP and apply the lifting procedure. If the obtained inequality is violated by the current fractional optimal solution, this inequality is added to (RSP) and we solve (RSP) again. These steps are repeated until no more violated inequalities can be found or an integer solution to (RSP) is found. When we cannot find any more violated inequalities and the optimal solution to (RSP) is still fractional, we start the branch-and-bound step. At each node of the enumeration tree, we apply the same procedure used at node 0.

When branching is needed at a node of the enumeration tree, we select a variable among variables y_j for $j \in J$ whose value is closest to 0.5. We do not need to branch on the variable x because if all values of y are integral in an optimal solution to (RSP), then x is also integral. This can be formalized by the following result.

Proposition 1: If all values of y are integral in an optimal solution to (RSP), then x is also integral.

Proof: Without loss of generality, suppose $(x_1, \ldots, x_{|F|}, y_1, \ldots, y_{|J|})$ be an optimal solution to (RSP). By the assumption, we can divide J into two distinct subsets J_1 and J_0 as $J_1 = \{j | y_j = 1 \text{ for } j \in J\}$ and $J_0 = \{j | y_j = 0 \text{ for } j \in J\} = J \backslash J_1$. If $j \in J_1$ ($y_j = 1$), it is clear that $x_f = 1$ for $f \in N_j$ by constraint (15). If $j \in J_0$ ($y_j = 0$), $0 \leq x_f \leq 1$ for $f \in N_j$. If the value of x_f for $f \in N_j$ is > 0 and < 1, x_f may be decreased to zero without violating constraints because all α_f's are strictly positive (see objective function 14). Thus, if $x_i \neq 0$ and $x_i \neq 0$ for $f \in N_j$, this solution is not an optimal solution. ■

Suppose that y_j is the variable on which we perform branching. Then, we make two new nodes in the enumeration tree, one with $y_j \leq 0$, the other with $y_j \leq 1$. Instead of adding those constraints explicitly to (RSP), we redefine the upper and lower bound of the variable. We use the best-bound rule (Nemhauser and Wolsey 1988) for node selection in the branch-and-bound procedure.

4 Overview of the algorithm for (MP)

This section gives a brief and overall explanation of our algorithm. First, we consider the overall procedure to solve (MP). Then, we consider the construction of initial columns needed to start the column-generation procedure to solve the LP relaxation of (MP). Finally, we consider a possible branching strategy to find an integer optimal solution for (MP).

4.1 *Overall procedure to solve (MP)*

It is assumed that a subset $G' \subset G$, which is the set of initial columns, is given. After we solve (RMP′), which is the linear programming problem obtained by replacing G with G' in (RMP), we get optimal dual values π_j, for all $j \in J$, corresponding to rows in constraints (11). Using these values, we solve the (SP). If the optimal objective value > 0, we construct a new entering column using the optimal solution to (SP).

This column is added to (RMP′), and the procedure is repeated until no columns are generated. If the optimal objective value to (SP) is not >0, no more columns need to be added to (RMP′). When no more columns are generated and if the optimal solution to the current (RMP′) is integral, we get an optimal solution to (MP). If not, we use the branch-and-bound algorithm to the final formulation. Though the proposed algorithm can not guarantee optimal solutions, the proposed algorithm can yield high-quality solutions and tight lower bounds at the same time. However, we can also try to find an integer optimal solution by using the branch-and-price algorithm instead of the branch-and-bound algorithm. A branching strategy compatible to the column generation is proposed in section 4.3.

4.2 *Construction of initial columns for (MP)*

This section presents a procedure to obtain initial columns. Consider two different PCBs j_1 and j_2 such that $N_{j_1} \subseteq N_{j_2}$. It can be easily shown that at least one optimal solution have PCB j_1 and PCB j_2 in the same group (Daskin *et al.* 1997). In this case, we say that the PCB j_2 dominates PCB j_1. The procedure to find initial columns starts with removing the dominated PCBs from the set of all PCBs. Let J' denote the resulting set of PCBs.

The PCB grouping problem is NP-hard, but Daskin *et al.* (1997) also showed that if each group is constrained to have two or fewer PCBs and no PCB is produced in more than a single group, the problem can be solved in polynomial time using a minimum weighted matching algorithm. Thus, we can construct a set of groups, G', having two or fewer PCBs from J' by using the minimum weighted matching algorithm. And then, for each pair of elements $g, h \in G'$, we define two numbers ϕ_{gh} and ρ_{gh}, which indicate similarity and dissimilarity between g and h, as follows.

ϕ_{gh} number of feeders used by both group g and group h,
ρ_{gh} number of feeders used by either group g only or group h only.

Using the two numbers ϕ_{gh} and ρ_{gh}, we define the similarity measure of two groups g and h as $\text{sim}_{gh} = \phi_{gh}/(\phi_{gh} + \rho_{gh})$. A pair of groups having the largest value of the similarity measure is merged into one group. After a new group is found, the similarity measures associated with the new group are redefined. This procedure is repeated until no more merged groups can be found. Above discussions to construct initial columns of (MP) are summarized in the following procedure.

Procedure 1: Heuristic algorithm to find initial columns:

Step 1: Remove all dominated PCBs:

1.1: Set $J' = J$
1.2: For all $j \in J$, if $N_j \subseteq N_k$, for $k \neq j$, set $J' \leftarrow J' \backslash \{j\}$.

Step 2: Construct G', having 2 or fewer PCBs, by using the minimum weighted matching algorithm from J'.
Define a set $R = \{(g,h) \big| g \in G', h \in G', \phi_{gh} + \rho_{gh} \leq B\}$.

Step 3: If $R = \phi$, stop.
Otherwise, find a pair of groups $(g^*, h^*) = \arg\max_{(g,h)\in R}(\text{sim}_{gh})$

And, let k^* be the new group obtained by merging g^* and h^*.

Step 4: Set $G' \leftarrow G' \backslash \{g^*, h^*\}$. And, set $G' \leftarrow G' \cup \{k^*\}$.
Redefine set R and go to Step 3.

4.3 *Possible branching strategy*

The present study has not incorporated the branching scheme in the proposed algorithm for (MP) since the RMP gives either integer optimal solutions or tight lower bounds for many cases in our computational study. However, we can also try to find an integer optimal solution by using the branch-and-price algorithm, which is similar to the branch-and-bound procedure except that we solve the subproblem at each node of the branch-and-bound tree by using the column generation.

When the branch-and-price approach is used, the main difficulty arises in the column generation after some of the variables are fixed at 0. To prevent the generation of columns that were set to 0, a careful branching rule should be used. For the current problem, we can use a branching scheme due to Ryan and Foster (Ryan and Foster 1981, Vance *et al*. 1994). The branching scheme consists of partitioning the set of solutions into those in which two specific PCBs lie in different groups, and those in which they lie in the same group.

Setting two specific PCBs h and j lie in different groups is equivalent to setting:

$$\sum_{g \in G(h,\, j)} \lambda_g = 0,$$

where $G(h,j)$ is the set of groups having PCB h and PCB j.

In this case, we set $\lambda_g = 0$ for all $g \in G(h, j)$ when solving (RMP) and set $y_h + y_j \leq 1$ when solving (SP). On the other hand, setting two specific PCBs h and j lie in the same group is equivalent to setting:

$$\sum_{g \in \bar{G}(h,\, j)} \lambda_g = 0$$

where $\bar{G}(h, j)$ is the set of groups having either PCB h or PCB j exclusively. In this case, we set $\lambda_g = 0$ for all $g \in \bar{G}(h, j)$when solving (RMP) and set $y_h = y_j$ when solving (SP).

The proposed branching scheme for (MP) was not implemented in our computational study since the branch-and-bound procedure only gave satisfactory results. However, if any one wants to find the optimal solution, the above-mentioned branching scheme can be used in the full branch-and-price algorithm.

5. Computational results

5.1 *Construction of the data set*

This section outlines the test results of the proposed algorithm for the four data sets: DATA1, DATA2, DATA3 and DATA4. DATA1 consists of 10 problems presented in previous research. Characteristics of problems in *DATA1* are summarized in table 1. Problem A1 is listed in Maimon and Shtub (1991). Problem A2 is listed in Hashiba and Chang (1991). Problem A5 is randomly generated by the same method proposed by Daskin *et al*. (1997). A3, A4 and A6–10 are randomly generated by the same method proposed by Bhaskar and Narendran (1996). Compared with our

Table 1. Test results for *DATA1*.

Problem number	Size ($\|J\| \times \|F\|$)	Capacity B	PCB Set-up cost	Set-up time		
				No grouping (PC, FC, TC)	Previous (PC, FC, TC)	Proposed (PC, FC, TC)
A1	8×53	15	10	(80, 78, 158)	(80, 68, 148)[a]	(80, 68, 148)
A2	9×20	14	10	(90, 74, 164)	(90, 58, 148)[b]	(90, 48, 138)
A3	9×30	20	6	(54, 79, 133)	(66, 48, 114)[d]	(54, 52, 106)
A4	15×50	30	10	(150, 197, 347)	(170, 120, 290)[d]	(150, 119, 269)
A5	16×53	12	10	(160, 113, 273)	(160, 76, 236)[c]	(160, 76, 236)
A6	18×65	35	15	(270, 345, 615)	(300, 242, 542)[d]	(270, 264, 534)
A7	23×60	35	10	(230, 387, 617)	(265, 244, 509)[d]	(230, 268, 498)
A8	25×75	45	15	(375, 557, 932)	(375, 411, 786)[d]	(375, 376, 751)
A9	28×90	55	15	(420, 783, 1203)	(435, 596, 1001)[d]	(420, 532, 942)
A10	30×100	60	20	(600, 926, 1526)	(600, 717, 1317)[d]	(600, 643, 1243)

Sources: [a]Maimon and Shtub (1991),
[b]Hashiba and Chang (1991),
[c]Daskin *et al.* (1997),
[d]Bhaskar and Narendran(1996).

study, Maimon and Shtub (1991) and Bhaskar and Narendran (1996) allowed each type of PCB to be contained in more than one group. However, because of the problem structure, it is more advantageous in terms of cost to include each PCB in one group. Therefore, their results were compared with the results of our algorithm. For all problems A1–10 in DATA1, we set the set-up time of each feeder to be 1 and the number of lanes occupied by each feeder also to be 1 in order to compare the performance of the proposed algorithm with the other algorithms presented in previous studies.

The problems in DATA2, DATA3 and DATA4 are constructed from problems in DATA1 by changing some input parameters. For the problems in DATA2, DATA3 and DATA4, we generated the set-up cost of each feeder, the number of lanes occupied by each feeder and the machine capacity randomly. For all problems in DATA2, DATA3 and DATA4, the set-up cost of each feeder and the number of stages occupied by each feeder were randomly generated from a discrete uniform [1, 10] and [1, 4], respectively. The machine capacity was generated as follows. For each problem, we found the number of lanes, denoted by $l(j)$, necessary to allocate all feeders for PCB j. We then multiplied a real number r by the maximum number among $l(j)$ for all $j \in J$. We used $r = 1.1$, 1.3 and 1.5 for the problems in DATA2, DATA3 and DATA4, respectively. Note that as r becomes large, the machine capacity becomes large. This makes the problem more complicated.

We used the CPLEX 4.0 callable library as the LP solution routine and the other routines for adding inequalities and changing bounds of variables. The test problems were solved on Pentium III (500-MHz) computer.

5.2. *Performance of the column-generation approach*

The results for 10 problems in DATA1 are summarized in table 1. The heading 'No grouping' refers to the set-up times (s) obtained by the simple procedure that performs the mounting operations on each PCB serially without PCB grouping. The heading 'Previous' refers to the set-up times (s) obtained by the algorithm from the

sources mentioned in table 1. The heading 'Proposed' refers to the set-up times (s) obtained by the proposed algorithm given in the study. The headings 'PC', 'FC' and 'TC' in parentheses refer to the sum of PCB set-up cost, the sum of feeder set-up cost, and the sum of both PCB set-up cost and feeder set-up cost, respectively.

Table 1 shows that the proposed column generation algorithm outperforms other PCB grouping algorithms proposed in previous research for all problems. Tables 2–5 summarize the test results to evaluate the proposed approach. The headings 'IP' and 'LP' refer to the best objective values of (MP) found by the algorithm and the optimal objective value of the LP relaxation of (MP), respectively. The heading 'GAP' refers to the relative ratio of the optimal objective value of the LP relaxation of (MP) to the best objective value of (MP) found by the algorithm, i.e. [(best objective value of (MP) – optimal objective value of LP relaxation)/best objective value of (MP)] × 100. The heading 'Number of B&B nodes' refers to the number of nodes generated in the branch-and-bound procedures. The headings 'Initial columns' and 'Additional columns' refer to the number of initial columns of master problem and the number of columns generated by the column generation problem (SP).

Table 2. Computational results for DATA1.

Problem number	Size ($\|J\| \times \|F\|$)	Capacity B	Solutions		GAP at node 0 (%)	Number of B&B nodes	Number of columns		Solution time (s)
			IP	LP			Initial	Additional	
A1	8 × 53	15	68	68.00	0.00	–	11	3	3.8
A2	9 × 20	14	48	48.00	0.00	–	12	6	5.2
A3	9 × 30	20	52	52.00	0.00	–	12	8	7.4
A4	15 × 50	30	119	119.00	0.00	–	20	25	103.1
A5	16 × 53	12	76	75.67	0.43	3	22	17	28.5
A6	18 × 65	35	264	264.00	0.00	–	27	15	149.8
A7	23 × 60	35	268	260.00	2.98	31	31	37	958.0
A8	25 × 75	45	376	369.90	1.62	18	34	41	1681.2
A9	28 × 90	55	532	527.20	0.90	5	39	36	2526.9
A10	30 × 100	60	643	643.00	0.00	–	42	45	6342.6

Table 3. Computational results for DATA2.

Problem number	Size ($\|J\| \times \|F\|$)	Capacity B	Solutions		GAP at node 0 (%)	Number of B&B nodes	Number of columns		Solution time (s)
			IP	LP			Initial	Additional	
B1	8 × 53	31	71	71	0.0	–	10	0	0.9
B2	9 × 20	35	55	54.3	1.27	3	13	4	4.5
B3	9 × 30	32	59	59	0.0	–	12	3	3.2
B4	15 × 50	51	170	170	0.0	–	20	6	21.5
B5	16 × 53	23	78	77.8	0.26	3	22	15	29.6
B6	18 × 65	65	313	313	0.0	–	22	8	54.9
B7	23 × 60	61	323	323	0.0	–	30	14	123.8
B8	25 × 75	70	504	504	0.0	–	31	4	56.6
B9	28 × 90	85	729	729	0.0	–	33	1	31.5
B10	30 × 100	101	811	811	0.0	–	38	10	225.1

Table 4. Computational results for DATA3.

Problem number	Size ($\|J\| \times \|F\|$)	Capacity B	Solutions IP	Solutions LP	GAP at node 0 (%)	Number of B&B nodes	Number of columns Initial	Number of columns Additional	Solution time (s)
C1	8×53	36	67	67	0.0	–	10	4	3.9
C2	9×20	42	41	41	0.0	–	12	8	6.1
C3	9×30	38	54	54	0.0	–	11	9	9.8
C4	15×50	60	140	138.5	1.07	1	21	12	45.4
C5	16×53	27	72	70.2	2.50	3	22	23	40.1
C6	18×65	77	275	275	0.0	–	27	9	66.9
C7	23×60	72	280	276	1.43	10	33	31	557.9
C8	25×75	83	425	422.5	0.59	1	35	21	346.4
C9	28×90	100	625	625	0.0	–	38	19	371.9
C10	30×100	120	704	700	0.58	5	43	28	2541.2

Table 5. Computational results for DATA4.

Problem number	Size ($\|J\| \times \|F\|$)	Capacity B	Solutions IP	Solutions LP	GAP at node 0 (%)	Number of B&B nodes	Number of columns Initial	Number of columns Additional	Solution time (s)
D1	8×53	42	65	63.3	2.61	3	11	7	7.2
D2	9×20	48	32	32	0.0	–	11	11	4.6
D3	9×30	44	49	49	0.0	–	12	10	10.9
D4	15×50	69	119	119	0.0	–	20	25	116.6
D5	16×53	32	67	65.8	1.79	1	21	30	60.4
D6	18×65	89	244	239.4	1.89	7	25	22	255.4
D7	23×60	83	246	229.7	6.62	14	30	43	1245.4
D8	25×75	96	379	372.9	1.82	20	34	42	1806.5
D9	28×90	116	550	542.5	0.15	4	39	31	1801.7
D10	30×100	138	601	592.9	1.35	22	40	50	11849.8

Finally, the heading 'Solution time (s)' refers to the execution times needed to solve the problems. Table 2 shows that we get the optimal solution at node 0 without branch-and-bound phase in six of 10 problems. An optimal solution was also found for one problem (problem 5) in the branch-and-bound phase. Moreover, the solutions are obtained within a reasonable time.

Tables 3–5 summarize the test results for the problems in DATA2, DATA3 and DATA4. We found optimal solutions for 18 of 30 problems.

6. Conclusions

This paper considered a PCB grouping problem to minimize the set-up time of component feeders. This problem is not new in the sense that many researchers have studied similar or the same problems (Hashiba and Chang 1991, Maimon *et al.* 1993, Daskin *et al.* 1997). Contrary to previous research, an

integer-programming approach was used to solve the PCB grouping problem. In this approach, the original problem was decomposed into a master problem and a column-generation subproblem. Starting with a few columns in the master problem, new columns were generated successively by solving the subproblem optimally. To solve the subproblem, the branch-and-cut approach was used. To solve the master problem, the branch-and-bound algorithm was used. The test results for a number of real-world problems and randomly generated problems show that the approach performs very well compared with other previous approaches in terms of solution quality. In terms of computing time, the approach may need more computation time than previous solution approaches, which are heuristic based. However, this application is not very sensitive to computation time since the grouping of PCBs is planned well ahead of actual production, and the computational results show that most of the problems can be solved within a reasonable time.

References

Bhaskar, G. and Narendran, T.T., Grouping PCBs for set-up reduction: a maximum spanning tree approach. *Int. J. Prod. Res.*, 1996, **34**, 621–632.

Boyd, E.A., Polyhedral results for the precedence constrained knapsack problem. *Disc. Appl. Math.*, 1993, **41**, 185–201.

Carmon, T.F., Maimon, O.Z. and Dal-El, E.M., Group set-up for printed circuit board assembly. *Int. J. Prod. Res.*, 1989, **27**, 1795–1810.

Crama, Y., Kolen, A.W.J., Oerlemans, A.G. and Spieksma, F.C.R., Throughput rate optimization in the automated assembly of printed circuit boards. *Ann. Oper. Res.*, 1990, **26**, 455–480.

Daskin, M.S., Maimon, O., Shtub, A. and Braha, D., Grouping components in printed circuit board assembly with limited component staging capacity and single card setup: problem characteristics and solution procedure. *Int. J. Prod. Res.*, 1997, **35**, 1617–1638.

Gilmore, P.C. and Gomory, R.E., A linear programming approach to the cutting-stock problem. *Oper. Res.*, 1961, **9**, 849–859.

Hashiba, S. and Chang, T., PCB assembly set-up reduction using group technology. *Comput. Ind. Eng.*, 1991, **21**, 453–457.

Johnson, E.L., Mehrotra, A. and Nemhauser, G.L., Min-cut clustering. *Math. Progr.*, 1993, **62**, 133–151.

Kusiak, A., The generalized group technology concept. *Int. J. Prod. Res.*, 1987, **25**, 561–569.

Luzzato, D. and Perona, M., Cell formation in PCB assembly based on production quantitative data. *Eur. J. Oper. Res.*, 1993, **69**, 312–329.

Maimon, O.Z., Dal-El, E.M. and Carmon, T.F., Set-up saving schemes for printed circuit board assembly. *Eur. J. Oper. Res.*, 1993, **70**, 177–190.

Maimon, O. and Shtub, A., Grouping methods for printed circuit board assembly. *Int. J. Prod. Res.*, 1991, **29**, 1379–1390.

Nemhauser, G.L. and Wolsey, L.A., *Integer and Combinatorial Optimization*, 1988 (Wiley, New York).

Padberg, M. and Rinaldi, G., A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Rev.*, 1991, **33**, 60–100.

Park, K. and Park, S., Lifting cover inequalities for the precedence-knapsack problem. *Disc. Appl. Math.*, 1997, **72**, 219–241.

Rajkumar, K. and Narendran, T.T., A heuristic for sequencing PCB assembly to minimize set-up times. *Prod. Plan. Control.*, 1998, **9**, 465–476.

Randhawa, S.U., McDowell, E.D. and Faruqui, S., An integer-programming application to solve sequencer mix problems in printed circuit board production. *Int. J. Prod. Res.*, 1985, **23**, 543–552.

Ryan, D.M. and Foster, B.A., An integer programming approach to scheduling, in *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling*, pp. 269–280, 1981, edited by A. Wren (North-Holland, Amsterdam).

Shtub, A. and Maimon, O., Role of similarity measures in PCB grouping procedures. *Int. J. Prod. Res.*, 1992, **30**, 973–983.

Sokal, R.R. and Sneath, P.H.A., *Principles of Numerical Taxonomy*, 1963 (San Francisco, Freeman).

Vance, P.H., Banhart, E.L., Johnson, E.L. and Nemhauser, G.L., Solving binary cutting stock problem by column generation and branch-and-bound. *Computat. Optim. Appl.*, 1994, **3**, 111–130.

Yu, S., Sohn, J., Park, S. and Oh, B.J., Efficient operation of a multi-functional surface mounting device. *Comput. Ind. Eng.*, 1997, **33**, 797–800.