한국경영과학회/대한산업공학회 2003 춘계공동학술대회
2003 년 5 월 16 일-17 일 한동대학교(포항)

An Integer Programming Approach to the PCB Grouping Problem

Sungyeol Yu[a], Duksung Kim[b], and Sungsoo Park[b]

[a]Division of Business Administration, Catholic University of Pusan,
Bugok3-dong, Kumjung-gu, Busan 609-757, Republic of Korea
Tel: 051-510-0663, Fax: 051-510-0668, e-mail: syyu@cup.ac.kr
[b]Department of Industrial Engineering, KAIST
Guseong-dong, Yuseong-gu, Daejeon 305-701, Republic of Korea

Abstract

We consider a PCB grouping problem arising from the electronic industry. Given a surface mounting device, several types of PCBs and a number of component feeders used to assemble the PCBs, the optimization problem is the PCB grouping problem while minimizing setup time of component feeders.

We formulate the problem as an integer programming model and propose a column generation approach to solve the integer programming formulation. In this approach, we decompose the original problem into master problem and column generation subproblem. Starting with a few columns in the master problem, we generate new columns successively by solving subproblem optimally. To solve the subproblem, we use a branch-and-cut approach. Computational experiments show that our solution approach gives high quality solutions in a reasonable computing time.

1 Introduction

The electronic industry relies heavily on the production rate of surface mounting devices (SMDs) to be used for the mounting of electronic components on the surface of printed circuit boards (PCBs) (Crama *et al.* 1990). The production rate in the PCB manufacturing environments depends on the two factors that influence time for the PCB assembly. One is the time needed to assemble a PCB when a PCB is produced (Crama *et al.* 1990, Yu *et al.* 1997). And, the other is the setup 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). In this paper, we consider an optimization problem concerned to the second case, which we call a PCB grouping problem.

The problem considered in this study can be described as follows. A set of several types of PCBs is to be produced on a SMD. The machine has limited lanes to which feeders are placed. The set of PCBs are divided into some groups satisfying the lane capacity of the machine. All component feeders, which are necessary to produce all PCBs included in each group, must be loaded into the machine simultaneously. In the general case, PCBs are allowed to be loaded multiple times (one per group) and components are allowed to be included in multiple groups. However, the proposed solution method will focus on a restricted class of problems in which each PCB is loaded exactly once. In that case, all of the component feeders required by a PCB must be included in the group in which the PCB is included. Thus, the problem can be defined to determine the set of PCB groups with the objective of minimizing the total setup cost of component feeders.

Now, we give some details of our problem including assumptions as follows.

1) The setup of a component feeder consists of loading and unloading into the machine stage.
2) A mechanical setup (i.e., changing the dimension of the machine's table or changing the width of the conveyor carrying the PCBs to the machine) between PCBs in the same production group is not required. This is usually accomplished by 'paneling', a method in which several PCBs are assembled as a single standard sized panel that is later cut to the correct dimensions.
3) PCB transfer times into and out of the machines are negligible.
4) Refilling components in the machines during assembly is not considered, since the quantity of each component required is independent of the scheduling method used.
5) The number of lanes required by all feeders is larger than the machine capacity.
6) Machine operation time is not significant.

The PCB grouping problem is not new in the sense that many researchers have studied similar or the same problems (Carmon *et al.*, 1989, Hashiba

and Chang, 1991, Maimon and Shtub, 1991, Luzzato and Peona, 1993, Daskin *et al.*, 1997).

Contrary to the previous researches based on heuristic algorithm, we propose a new approach to solve the PCB grouping problem by a column generation approach. In our approach, we decompose the original problem into master problem and column generation subproblem. Starting with a few columns in the master problem, we generate new columns successively by solving subproblem optimally. The process of adding columns is repeated until no more profitable column can be found. To solve the subproblem optimally, we use a 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 remainder of this paper is organized as follows. In section 2, we formulate the PCB grouping problem as two integer linear programming problems. In section 3, we propose a branch-and-cut algorithm to solve the column generation problem. In Section 4, we provide the overall solution approach to solve the PCB grouping problem. In section 5, we summarize our computational experiences with the proposed algorithm to the some real world problems and some randomly generated problems. Finally, concludings are given in section 6.

2 Mathematical Formulations

We present an integer programming formulation for the PCB grouping problem. 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 restricted number of columns that indicate some grouping configurations and the subproblem is constructed to generate columns entered into the master problem. We define the following notation used in this paper.

Notation

F: set of component feeders

J: set of PCB types

α_f : setup cost of feeder f

B: machine capacity (the number of lanes in the machine)

s_f : the number of lanes occupied by feeder f

$N_j = \{f | a_{fj} = 1\}$: set of indices of feeders required by PCB j, where $a_{fj} = 1$, if components in feeder f is needed by PCB j, 0, otherwise

G: the set of all possible PCB grouping configurations

$W(g)$: the set of indices of feeders required by a PCB group configuration $g \in G$

$p_{jg} = 1$ if PCB j is included in a grouping configuration g, 0 otherwise.

Note that an element $g \in G$ must be satisfied by the inequality $\sum_{f \in W(g)} s_f \le B$. We define a decision variable as follows.

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

With these notation and decision variables, the PCB grouping problem may be formulated as follows:

$$\text{(MP)} \min \sum_{g \in G} c_g \lambda_g \tag{1}$$

$$\text{s.t} \sum_{g \in G} p_{jg} \lambda_g = 1, \text{for all } j \in J, \tag{2}$$

$$\lambda_g \in \{0,1\}, \text{for all } g \in G \tag{3}$$

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

The objective function (1) states the sum of costs of PCB-grouping configuration selected. Constraints (2) ensure that each PCB must be included in exactly one group.

The problem (MP) have exponentially many columns in case that the number of PCBs and feeders are large. However, we can solve a relaxation problem of (MP) efficiently with a few columns by using a column generation technique. Let (RMP) be the LP relaxation problem of (MP). Now, we assume that a subset $G' \subset G$ of grouping configuration is given. We define the restricted linear program (RMP′) replacing G by G' in (RMP). The solution of (RMP′) is suboptimal to (RMP). Let π_j be the dual variable associated with j-th constraint in (2). Then, the constraints in the dual problem of (RMP) are

$$\sum_{j \in J} p_{jg} \pi_j \le c_g, \text{for all } g \in G. \tag{4}$$

Let π^* be an optimal solution to the dual of (RMP′). Then, it is also optimal to the dual of (RMP) if $\sum_{j \in J} p_{jg} \pi_j^* \le c_g$, for all $g \in G \setminus G'$. From (4), the optimality condition for (RMP) is

$$\max\left(\sum_{j \in J} p_{jg} \pi_j^* - \sum_{f \in W(g)} \alpha_f \,\middle|\, g \in G \right) \le 0 \tag{5}$$

Thus, we can solve (RMP) staring with some

restricted initial columns by introducing columns that satisfies the optimality condition (5) continuously until no more columns are generated.

To formulate the column generation problem, we introduce variables x_f and y_j, which indicate whether feeder f and PCB j are included the group or not, respectively. If PCB j is included in a group ($y_j = 1$), then corresponding feeders are included in the group ($x_f = 1$ for $f \in N_j$). And, the number of lanes occupied by all feeders included in the group equals to or less than the machine capacity, B. Thus, the column generation problem can be formulated as follows.

$$\text{(SP) max} \quad \sum_{j\in J} \pi_j y_j - \sum_{f\in F} \alpha_f x_f \tag{6}$$

$$\text{s.t.} \quad y_j \le x_f, \text{ for all } f \in N_j \text{ and } j \in J \tag{7}$$

$$\sum_{f\in F} s_f x_f \le B \tag{8}$$

$$y_j \in \{0,1\}, \text{for all } j \in J \tag{9}$$

$$x_f \in \{0,1\}, \text{for all } f \in F \tag{10}$$

Constraints (7) imply that if any PCB j is included in a group, all feeders needed to assemble the PCB must be also included in the group. Constraint (8) states that the number of lanes occupied by all feeders included in a group equals to or less than the number of machine capacity.

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

3. Subproblem Optimization

3.1 Valid inequality

The problem (SP) is a generalization of the knapsack problem by including 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, only if x_f equals to zero for $f \in N_j$, then the corresponding y_j equals to zero.

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 $y_j \le x_f$, for all $f \in N_j$ and $j \in J$. For $(f, j) \in A$, node f is called a *predecessor* of node j and node j is called a *successor* of node f.

The pair of node $k_1 \in V$ and node $k_2 \in V$ 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 pair wise incomparable. Note that the sets F and J, both are incomparable.

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 a called a cover if $\sum_{f\in C} s_f > B$ (Nemhauser and Wolsey 1988). And the associated inequality $\sum_{f\in C} x_f \le |C| - 1$ is called 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 $\sum_{f\in C} x_f \le |C| - 1$ is valid for the knapsack polytope. However, when there exist precedence constraints between variables as our application, the following modification is more useful. We give the following notation throughout remainder of this paper.

$$N(C) = \bigcup_{j\in C} N_j$$

$$T(C) = C \cup N(C)$$

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

(i) C is incomparable

(ii) $\sum_{f\in T(C)} s_f > B$, and

(iii) $\sum_{f\in T(C\setminus\{j\})} s_f \le B$, for all $j \in C$

In words, a minimal induced cover is a set of incomparable items, which together do not fit in the knapsack, whereas all but one of them does fit in the knapsack together. The 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 polytope of the set of constraints in (SP). We refer to this inequality as the MIC-inequality.

$$\sum_{j\in C} y_j \le |C| - 1 \tag{11}$$

With the MIC-inequality, We can get stronger cutting planes by lifting the variables in F and $J \setminus C$ into the MIC-inequality using a lifting procedure. Lifting is a systematic procedure to obtain valid inequalities for a polyhedron from valid inequalities for lower dimensional polyhedron. For the details of the general lifting procedure, refer

Nemhauser and Wolsey (1988) and Park and Park(1997).

3.2 Separation

Now, we give a formulation of the separation problem to find a minimum induced cover and an algorithm to solve the problem. Given a fractional solution (x^*, y^*) of (SP), the problem is to find a minimum induced cover C satisfying the following inequality.

$$\sum_{j\in C} y_j^* > |C| - 1 \tag{12}$$

Inequality (12) is equivalent to $\sum_{j\in C}(1-y_j^*) < 1$. All values of s_f for $f \in F$ are positive integers. Thus, the separation problem can be formulated as follows.

$$\text{(SEP)} \quad \min \sum_{j\in J}(1-y_j^*)z_j \tag{13}$$

$$\text{s.t.} \sum_{f\in F} s_f x_f \geq B+1 \tag{14}$$

$$x_f \leq \sum_{\{j:\, f\in N_j\}} z_j, \quad \text{for all } f \in F \tag{15}$$

$$x_f \in \{0,1\}, \quad \text{for all } f \in F \tag{16}$$

$$z_j \in \{0,1\}, \quad \text{for all } j \in J \tag{17}$$

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

Now we give a heuristic algorithm to solve (SEP). The basic idea of the heuristic algorithm is to add an element in J, having the more successors and the larger value of y_j^*, to C continuously until the knapsack is filled with the feeders, updating the current precedence-successor graph. Given an associated graph $D = (V, A)$, the heuristic algorithm is described as follows.

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 a graph $D^t = (V^t, A^t)$,

Compute the weight

$u_j^t = \sum_{f\in N_j^t} s_f / (1-y_j^*)$ for $j \in J^t$.

Set $j^* = \arg\max_{j\in J^t} \{u_j^t\}$

(if ties are happened, select arbitrary).

Step 2: Update set C.

Add j^* to the C, i.e., $C \leftarrow C \cup \{j^*\}$.

Step 3: Check feasibility.

If $\sum_{\{f | f\in N_j^t, j\in C\}} s_f > B$, go to phase 2.

Otherwise, go to next step.

Step 4: Update graph.

Set $t \leftarrow t+1$.

$F^t = F^{t-1} \setminus \{f \in F^{t-1} \mid f \in N_{j^*}^{t-1}\}$.

$J^t = J^{t-1} \setminus \{j^*\}$.

$V^t = (F^t, J^t)$.

$A^t = A^{t-1} \setminus \{(f,j) \in A^{t-1} \mid f \in N_{j^*}^{t-1}\}$

$D^t = (V^t, A^t)$.

$N_j^t = \{f \in F^t \mid (f,j) \in A^t\}$.

Go to step 1.

Phase 2: Select minimal elements in C.

Without loss of generality, let the set C be $\{1,2,\cdots,|C|\}$.

For $k = 1$ to $|C|$, do

If $\sum_{f\in C\setminus\{k\}} s_f > B$, set $C \leftarrow C \setminus \{k\}$

3.3 Branch-and-cut procedure

Let (RSP) be the LP relaxation of (SP). The branch-and-cut procedure begins with 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 of the optimal solution to (SP) is provided by this solution. And we proceed to find a lifted minimal induced cover by the separation heuristic *HSEP* and the lifting procedure. This inequality is added to (RSP) and we solve (RSP) again. These steps are repeated until no more inequalities can be found or an integer solution to (RSP) is found. When we cannot find any more minimal induced covers, we start branch-and-bound step. At each node in the enumeration tree, we apply the same procedure used in node 0.

When branching is needed at any node in the enumeration tree, we select a variable among variables y_j for $j \in J$ whose absolute value of current value minus 0.5 is minimum. We need not to branch the variable x because if all values of y are integral in an optimal solution to (RSP), then x is also integral.

Assume that y_j is a branching variable on which we perform branching. Then, we make two new nodes in the enumeration tree, one with

$y_j \le 0$, the other with $y_j \ge 1$. Instead of adding those constraints explicitly to (RSP), we redefine the upper and lower bound of the variable. After a branch, we use best-bound rule (Nemhauser and Wolsey 1988) for node selection.

4. Algorithm for (MP)

4.1 Overall procedure to solve (MP)

We assume 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 of replacing G into G' in (RMP), we get optimal dual values π_j, for all $j \in J$, corresponding to rows in constraints (2). Using these values, we solve the (SP). If the optimal objective value is not satisfied by the optimality condition, we construct new entering column using the optimal solution value. This column is added to (RMP′), and the procedure is repeated until no columns are generated. When no more columns are not generated, if the optimal solution of the current (RMP′) is integral, we get an optimal solution to (MP). If not, we start a branch-and-bound algorithm to (MP). The overall procedure to solve (MP) is summarized below.

Procedure to solve (MP)

Step 1: Generate initial columns using the (Procedure 1) and construct initial LP relaxation problem (RMP′)

Step 2: Solve (RMP′) and get optimal dual values π_j for all $j \in J$.

Step 3: Construct the column generation problem (SP) using π_j, and solve the (SP). If an entering column is generated, introduce the column to (RMP′) and go to step 2. Otherwise, go to step 4.

Step 4: If an optimal solution of the current (RMP′) is integral, we get an optimal solution to (MP) and the procedure is terminated. Otherwise, perform branch and bound algorithm.

4.2 Construction of initial columns for (MP)

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 fine initial columns starts from 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 2 or fewer PCBs and no PCB is produced 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 2 or fewer PCBs from J' by using the minimum weighted matching algorithm. And then, for each pairs 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}; the number of feeders used by both group g and group h.

ρ_{gh}; the 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 $sim_{gh} = \phi_{gh} / (\phi_{gh} + \rho_{gh})$. A pair of groups having the largest value of the similarity measure is merged into one group. After a new group is founded, the similarity measures associated with the new group are redefined. This procedure is repeated until more merged groups cannot be found.

4.3 Branching Strategy

In our study, we have not incorporated the branching scheme in the proposed algorithm since the RMP gives either integer optimal solutions in many cases or tight lower bound in the other cases. However, we can also try to find an integer optimal solution by using the branch-and-price algorithm, which is very similar to the branch-and-bound procedure except that we solve the subproblem at each node in 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 subset of the variables is 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 in partitioning the set of solutions into those in which two specific PCBs lie on different groups, and those in which they lie on the same group. Following shows the details.

Setting two specific PCBs h and j lie on 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 $y_h + y_j \le 1$ when solving (SP) and set $\lambda_g = 0$ for all $g \in G(h, j)$ when solving (MP). On the other hand, setting two specific PCBs h and j lie on the same group is

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

5 Computational Results

In this section, we outline test results of the proposed column generation approach to four data sets, *DATA1*, *DATA2*, *DATA3* and *DATA4*. *DATA1* consists of 10 problems presented in previous researches. The problems in *DATA2*, *DATA 3* and *DATA4* are constructed from problems in *DATA1* by changing some input parameters. For problems in *DATA1*, we assume that the setup time of each feeder equals to 1 and the number of lanes occupied by each feeder is also 1 in order to compare the performance of the proposed algorithm with the one of other algorithms presented in previous studies.

For the problems in *DATA2*, *DATA 3* and *DATA4*, we generate the setup cost of each feeder, the number of lanes occupied by each feeder and the machine capacity randomly. For all problems in *DATA2*, *DATA 3* and *DATA4*, the setup cost of each feeder and the number of stages occupied by each feeder are randomly generated from a discrete uniform [1, 10] and [1, 4], respectively.

The machine capacity is generated as follows. For each problem, we find the number of lanes, denoted by $l(j)$, necessary to allocate all feeders for PCB j. And then, we multiply a real number r by the maximum number among $l(j)$ for all $j \in J$. We use $r = 1.1$, $r = 1.3$ and $r = 1.5$ for the problems in *DATA2*, *DATA3* and *DATA4*, respectively. Note that the more the value of r is large, the more the size of knapsack is large. This makes the problem more complicate.

We used 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 are solved on Pentinum-3 (500 MHz).5.2 Performance of the column generation approach

The results for 10 problems in DATA1 are summarized in table 1. In table 1, we can see that the proposed column generation algorithm outperforms than other heuristic PCB grouping method proposed in previous researches for all problems. Furthermore, in table 2, we summarize results of performance measures to evaluate the proposed approach. In table 2, we can see that we get the optimal solution at node 0 without branch.

And, we found an optimal solution for one problem (problem no. 5) in the branch-and-bound phase. These are very attractive results in comparison with and-bound phase in 6 problems out of 10 problems. the ones by other approaches presented recently. Moreover, the solutions are obtained within reasonable time.

In table 3, table 4 and table 5, we summarize test results for the problems in *DATA2*, *DATA3* and *DATA4*. We found optimal solution for 18 problems of 30 problems.

Table1. Test results for *DATA1*

Problem No.	Size ($\|J\|\times\|F\|$)	Capacity B	Setup Time		
			Conventional	Heuristics	Proposed
A1[a]	8×53	15	78	68	68
A2[b]	9×20	14	74	52	48
A3	9×30	20	79	60	52
A4	15×50	30	197	140	119
A5[c]	16×53	12	113	76	76
A6	18×65	35	345	*N/A*	264
A7	23×60	35	387	*N/A*	268
A8	25×75	45	557	411	376
A9	28×90	55	783	581	532
A10	30×100	60	926	717	643

Remarks; a) source: Maimon and Shtub (1991), b) source: Hashiba and Chang (1991)
c) source: Daskin et al. (1997), source for others: Bhaskar and Narendran(1996)

Table 2. Performance analysis of the column generation approach for *DATA1*

Problem No.	Size ($\|J\| \times \|F\|$)	Solutions		GAP[1] at node 0 (%)	No. of B&B nodes	No. of columns		Solution Time (sec)
		IP	LP			Initial columns[2]	Additional Columns[3]	
A1	8×53	68	68	0.0	-	11	3	3.8
A2	9×20	48	48	0.0	-	12	6	5.2
A3	9×30	52	52	0.0	-	12	8	7.4
A4	15×50	119	119	0.0	-	20	25	103.1
A5	16×53	76	75.7	0.43	3	22	17	28.5
A6	18×65	264	264	0.0	-	27	15	149.8
A7	23×60	268	260.0	2.98	31	31	37	958.0
A8	25×75	376	369.9	1,62	18	34	41	1681.2
A9	28×90	532	527.2	0.90	5	39	36	2526.9
A10	30×100	643	643	0.0	-	42	45	6342.6

Remarks; 1) GAP: The ratio of the objective value of the LP relaxation to the optimal integer solution of (IP)
[(IP optimal – optimal value of LP relaxation) / IP optimal] × 100
2) Initial columns: number of initial columns of master problem
3) Additional columns: number of columns generated by the column generation problem (SP)

Table 3 Performance analysis of the column generation approach for *DATA2*

Problem No.	Size ($\|J\| \times \|F\|$)	Solutions		GAP at node 0 (%)	No. of B&B nodes	No. of columns		Solution time (sec)
		IP	LP			Initial columns	Additional columns	
B1	8×53	71	71	0.0	-	10	0	0.9
B2	9×20	55	54.3	1.27	3	13	4	4.5
B3	9×30	59	59	0.0	-	12	3	3.2
B4	15×50	170	170	0.0	-	20	6	21.5
B5	16×53	78	77.8	0.26	3	22	15	29.6
B6	18×65	313	313	0.0	-	22	8	54.9
B7	23×60	323	323	0.0	-	30	14	123.8
B8	25×75	504	504	0.0	-	31	4	56.6
B9	28×90	729	729	0.0	-	33	1	31.5
B10	30×100	811	811	0.0	-	38	10	225.1

Table 4. Performance analysis of the column generation approach for *DATA3*

Problem No.	Size ($\|J\| \times \|F\|$)	Solutions		GAP at node 0 (%)	No. of B&B nodes	No. of columns		Solution time (sec)
		IP	LP			Initial columns	Additional columns	
C1	8×53	67	67	0.0	-	10	4	3.9
C2	9×20	41	41	0.0	-	12	8	6.1
C3	9×30	54	54	0.0	-	11	9	9.8
C4	15×50	140	138.5	1.07	1	21	12	45.4
C5	16×53	72	70.2	2.50	3	22	23	40.1
C6	18×65	275	275	0.0	-	27	9	66.9
C7	23×60	280	276	1.43	10	33	31	557.9
C8	25×75	425	422.5	0.59	1	35	21	346.4
C9	28×90	625	625	0.0	-	38	19	371.9
C10	30×100	704	700	0.58	5	43	28	2541.2

Table 5 Performance analysis of the column generation approach for *DATA4*

Problem No.	Size ($\|J\| \times \|F\|$)	Solutions IP	Solutions LP	GAP at node 0 (%)	No. of B&B nodes	No. of columns: Initial columns	No. of columns: Additional columns	Solution time (sec)
C1	8×53	65	63.3	2.61	3	11	7	7.2
C2	9×20	32	32	0.0	-	11	11	4.6
C3	9×30	49	49	0.0	-	12	10	10.9
C4	15×50	119	119	0.0	-	20	25	116.6
C5	16×53	67	65.8	1.79	1	21	30	60.4
C6	18×65	244	239.4	1.89	7	25	22	255.4
C7	23×60	246	229.7	6.62	14	30	43	1245.4
C8	25×75	379	372.9	1.82	20	34	42	1806.5
C9	28×90	550	542.5	0.15	4	39	31	1801.7
C10	30×100	601	592.9	1.35	22	40	50	11849.8

6 Conclusions

In this paper, we consider a PCB grouping problem to minimize the setup time of component feeders. Contrary to previously researches, in this study, we use an integer programming approach to the PCB grouping problem. In this approach, we decompose 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 subproblem optimally. To solve the subproblem, we use a branch-and-cut approach. The test results for a number of real world problems and randomly generated problems show that our approach performs very well compared to other previous approaches.

References

Bhaskar, G. and Narendran, T. T., 1996, Grouping PCBs for set-up reduction: a maximum spanning tree approach, *International Journal of Production Research* 34, 621-632.

Boyd. E.A., 1993, Polyhedral results for the precedence constrained kanpsack problem, *Discrete Applied Mathematics* 41, 185-201.

Carmon, T. F., Maimon, O. Z. and Dal-El. E. M., 1989, Group set-up for printed circuit board assembly, *International Journal of Production Research* 27, 1795-1810.

Crama, Y., Kolen, A. W. J., Oerlemans, A.G. and Spieksma, F. C. R., 1990, Throughput Rate Optimization in the Automated Assembly of Printed Circuit Boards, *Annals of Operations Research* 26, 455-480

Daskin, M. S., Maimon, O., Shtub, A. and Braha, D., 1997, Grouping components in printed circuit board assembly with limited component staging capacity and single card setup: problem characteristics and solution procedure, *International Journal of Production Research* 35, 1617-1638.

Gilmore, P. C. and Gomory, R. E., 1961, A Linear Programming Approach to the Cutting-stock Problem, *Operations Research* 9, 849-859.

Hashiba, S. and Chang, T., 1991, PCB Assembly Setup Reduction Using Group Technology, *Computers and Industrial Engineering* 21, 453-457.

Kusiak, A., 1987, The Generalized Group Technology Concept, *International Journal of Production Research* 25, 561-569.

Maimon, O. Z., Dal-El. E. M. and Carmon, T. F., 1993 Set-up saving schemes for printed circuit board assembly, *European Journal of Operational Research* 70, 177-190

Maimon, O. and Shtub, A., 1991, Grouping Methods for Printed Circuit Board Assembly, *International Journal of Production Research* 29, 1379-1390.

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

Park, K. and Park, S., 1997, Lifting cover inequalities for the precedence-knapsack problem, *Discrete Applied Mathematics* 72, 219-241.

Randhawa, S. U., McDowell, E. D. and Faruqui, S., 1985, An integer programming application to solve sequencer mix problems in printed circuit board production, *International Journal of Production Research* 23, 543-552.

Ryan, D.M. and Foster, B.A., 1981, An Integer Programming Approach to Scheduling, In A. Wren (ed), *Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling* (Amsterdam: North-Holland), pp. 269-280

Shtub, A. and Maimon, O., 1992, Role of similarity measures in PCB grouping procedures, *International Journal of Production Research* 30, 973-983.

Yu, S., Sohn, J., Park, S. and Oh, B. J., 1997, Efficient Operation of a Multi-functional Surface Mounting Device, *Computers and Industrial Engineering* 33, 797-800