

Optimal Machine Setup and Product Clustering for Printed Circuit Board Manufacturing

Michael J. Magazine
Department of QAOM
University of Cincinnati
Cincinnati, Ohio 45221
Michael.Magazine@UC.edu

George G. Polak
MSIS Department
Wright State University
Dayton, Ohio 45435
George.Polak@Wright.edu

August 2, 2002

Abstract

In the manufacture of printed circuit boards, electronic components are attached to a blank board by one or more pick-and-place machines. Each machine has component magazines that house replaceable feeders. Since component requirements differ among board types, each change of feeders can improve processing time but indicates a machine setup. We develop an integer programming model for the joint problem of product clustering and machine setup. It is applicable to instances encountered in an electronic assembly environment. A numerical study is included.

Introduction

The production of Printed Circuit Boards (PCB) plays a crucial role in the contemporary electronics industry, and at the same time provides many challenges to modeling and optimization. The scheduling of jobs in a PCB shop, the specification of machine setups, and the sequencing of component insertions for a given job are among the topics that have attracted the attention of researchers. We consider the particular problem of jointly optimizing machine setup and PCB clustering. We develop an integer programming model for this problem, and investigate the computational capabilities of commercially available software to solve this model. Although the general case is NP-hard, two special cases common in practice are identified and shown to be easily solved.

There is an extensive literature on PCB operations. Ball and Magazine (1988) formulated a type of directed postman problem to determine the best sequence of insertion operations and developed an algorithm that is exact under certain conditions, and approximate within constant performance bounds when these conditions are relaxed. Hashiba and Chang (1991) formulated an integer program to minimize the number of setups over all sequences of jobs in a PCB assembly shop. In lieu of an exact algorithm, they proposed and tested a three-stage heuristic. Sadiq, Landers and Taylor (1993) presented and tested a two-stage heuristic known as the *intelligent slot assignment* algorithm to minimize total manufacturing time. The first stage sequences jobs to minimize setup time, and given this sequence, the second assigns components to sleeves to minimize processing time.

Jain, Johnson and Safai (1996) developed a nonlinear integer model for sequencing jobs in order to minimize setup time, and obtained approximate solutions

using a suite of four heuristics. In shop floor testing at Hewlett-Packard facilities, these solutions exhibited a tradeoff between setup time and processing time: for large batches, setup time reduction was surpassed by increased processing time. Assembly operations at Hewlett-Packard also motivated a study by Hillier and Brandeau (1998), who proposed a model for optimally assigning PCB types and components to manual processes as well as to machines. They developed a heuristic that provides near optimal solutions. Günther, Grunow and Schorling (1997) proposed a highly aggregated linear programming model to maximize system throughput in a high mix, low volume facility. To lessen the error of aggregation, the authors used a fuzzy estimation of the number of setups.

Cohn, Magazine and Polak (2000) formulated a set partitioning integer program with an exponential number of columns, which they called the Integrated Clustering and Machine Setup (ICMS) model, for the joint problems of product clustering and machine setup. In the prior literature, either the joint problem had been treated heuristically, or the two subproblems had been solved independently in sequential stages of a hierarchical planning scheme. Both approaches, in general, result in a solution that is suboptimal for the joint problem. Thus integration of these closely linked problems into a single model represented an improvement in PCB production planning. Magazine and Polak (2002) proposed a constraint programming approach to ICMS, and considered the relationship between the special case considered below in Section 2.3, and the general model in Section 3. Other work on ICMS has centered on the development, application, and testing of heuristics. Magazine, Polak and Sharma (2002) proposed and tested a heuristic for ICMS that employs a multi-exchange neighborhood search, and showed that this joint

problem is NP-hard. Norman (2001) devised a set of test problems and evaluated several classes of heuristics, including clustering by various metrics and genetic algorithms.

In this work, we propose an alternative and original integer programming formulation for ICMS. Our formulation is amenable to solution, for small instances, by off-the-shelf software. It does not comprise an exponential number of columns, and therefore does not require branch and price or column generation for small models. After describing the PCB assembly process, we review the underlying combinatorial optimization problem. We treat two special cases that occur in practice, and then develop an integer programming formulation for the general case. A numerical study and a preview of future research follow. The Appendix details a small-scale numerical illustration.

1. PCB Assembly

Ball and Magazine (1988) identified six principal steps in PCB assembly: (i) *production order release*, from materials requirement planning; (ii) *kitting*, the organizing of kits of appropriate components; (iii) *prepping*, the operations preparatory to insertion; (iv) *insertion* of components into PCB units; (v) inspection and *soldering* of components; and (vi) *testing* of PCB units for defects, failure, and functionality. Our focus falls on steps (iii) and (iv).

1.1 Pick and Place Operations

Prepping primarily consists of product clustering, i.e. the partitioning of the set of PCB types, and machine setups. Insertion operations are performed on a pick-and-place

machine that uses either through-hole or surface mount technology. In the former, a substrate (blank board) is pre-drilled, and a component pin is inserted into each hole. Solder circuitry is then applied to underside of the substrate. In surface mounting, solder paste is applied to the substrate and each component placed appropriately. The solder paste liquefies in an oven, yielding a finished board when cooled.

Figure 1 illustrates essential features of the particular pick-and-place machine used in the PCB assembly plant that motivated this study. It has a component magazine known as a feederbank, which is a linear array of sleeves or slots used to store the various components needed in the manufacture of PCB units. An insertion head moves from its home position at the midpoint of the array to pick an appropriate component from a sleeve and then returns to the home position to insert into a PCB. The board itself moves to the correct position during this head movement and does so quickly enough not to be an issue in this study. These operations are all numerically controlled. Though this technology is neither as new nor as fast as that used for large batch fabrication units, or "fabs," it is still common today, especially in high voltage or high amperage applications.

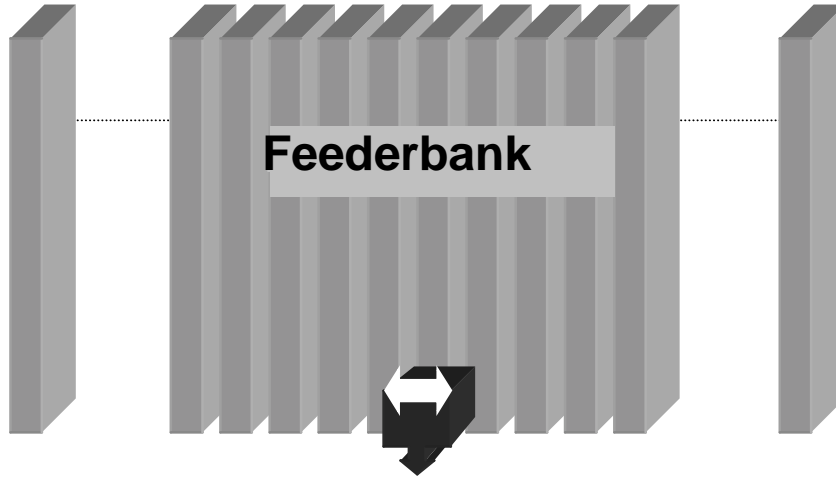


Figure 1. A pick-and-place machine. The insertion head is shown in home position at the midpoint of the feederbank.

1.2 Definitions and Assumptions

We now introduce terms and assumptions of our optimization model. *Total manufacturing time* for a set of jobs consists of the sum of *processing time* and *setup time*. Processing time for a PCB is defined to be the total time spent by the pick-and-place machine in component delivery and insertion. Each delivery and insertion from sleeve j requires d_j seconds and consists of moving the insertion head to that sleeve to pick a component, returning it to the home position, and placing the component on the board. Next, a *setup* is defined to be an assignment of components by type to feederbank sleeves such that each type of component is assigned to exactly one sleeve and that each sleeve contains exactly one type of component. (This one to one correspondence is always appropriate given the technology under consideration.) Each setup takes σ units of time, independent of the choice of assignment, and independent of any prior assignment. As discussed below, each setup is “full”, and no “partial” setups of the type

described by Jain et al(1996) are considered. Each type k of board has a distinct component profile, that is, any PCB of type k requires r_i^k of component type i , and is processed in a batch of known size b^k . By *job k* we mean the processing of b^k units of board type k , and all parameters are assumed to be nonnegative.

The assumptions listed here are based on our experiences in through-hole fabs, but hold reasonably well in the more complicated surface mount environment:

- (1) The time to pick and place is independent of placement sequence. Because the substrate board is small relative to the feeder bank, and the distances involved minute, positioning for placement is relatively quick, while the pick is the bottleneck operation.
- (2) setup times are independent of the partition of jobs into clusters. As in many PCB assembly operations, all setups we observed in the fabs were “full tear-down.” That is, all feeders were removed from the bank at the end of each run, and this activity is quite invariant in time or cost. The full tear-down setup is widely practiced for several reasons: it allows optimal placement of components for each cluster of jobs, it reduces opportunities to make mistakes, and it allows restocking of all feeders at once. Johnson (1995) documents its use in surface mount operations at Hewlett-Packard:

At the end of each batch, the operators would first take all of the feeders off of the machine and return them to the component storage area. Then the feeders for the new board were retrieved and installed on the machine. [The project manager] argued that removing all of the feeders was wasteful, since the new board often required some of the same components as the previous board. However, the mechanical engineer had overridden her objections. He pointed out that by tearing down the machine they

could optimize the placement of the feeders thus reducing the processing time...The production supervisor also argued that it was less likely for the operators to make mistakes in the setup using the tear-down technique...

(3) The feederbank sleeves are uncapacitated. For many machines, to reload a sleeve is a simple operation requiring very little down time, while other machines are equipped with bulk feeders appropriate for many types of components. In practice, sleeve capacity is quite flexible. Moreover, the full tear-down mode of setup affords an opportunity to restock all sleeves so that component “stockouts” are a rare occurrence during processing, as Johnson (1995) further notes:

With the tear-down techniques, the operator would usually ensure that there were enough components on each reel to run an entire batch.

(4) The number of distinct components is equal to the number of sleeves. There is no loss of generality due to this assumption: if the number of sleeves exceeds the number of components, then some sleeves would be left empty. On the other hand, suppose that the number of component types is actually larger than the number of sleeves. Among the component types, there is typically a subset common to all or most assembly jobs. The practice we observed in the fabs and commonly employed elsewhere is to use one pick and place machine, e.g., a high-speed machine, to process these insertions. Remaining “odd” components, often of unusual shape or size, are inserted manually, or inserted by a flexible pick and place machine as a follow up operation, see e.g. Hillier and Brandeau (1998).

2. The Integrated Clustering and Machine Setup Model

2.1 Formulation as a Combinatorial Optimization Problem

Intuitively, a machine setup that minimizes the processing time for a batch of several jobs may not minimize the processing time for any individual job. It follows that frequent setups, though time consuming, can reduce overall processing time. This fundamental tradeoff is observed in the fabs and has been documented, e.g. by Jain et al (1996); it moreover drives the formulation of our model to optimize total manufacturing time. Consider a set \mathbf{K} of jobs, each comprising a batch of a particular type of PCB assembled from a set of \mathbf{N} types of components in varying amounts. The pick-and-place machine has a set of \mathbf{N} uncapacitated sleeves, each of which can hold exactly one type of component. A machine setup consists of an assignment of component types to sleeves, i.e. a matching from \mathbf{N} to \mathbf{N} , and has a constant time of σ . The time to process all components of type $i \in \mathbf{N}$ used by job $k \in K$ from sleeve $j \in \mathbf{N}$ is given by $c_{ij}^k = b^k d_j r_i^k$. The Integrated Clustering and Machine Setup Model (ICMS) seeks to partition \mathbf{K} into subsets called *clusters* and determine a common setup for each cluster, so as to minimize the total manufacturing time. Magazine, Polak and Sharma (2002) gave the following formulation of ICMS as combinatorial optimization problem, and established it as NP-hard:

$$\text{ICMS Minimize } \{\sigma|F| + \sum_{k \in K} \sum_{i \in N} \sum_{j \in N} c_{ij}^k : (i, j, k) \in F\}. \quad (1)$$

The feasible set is $F = \bigcup_{k \in K} F(k)$, where for each job k , $F(k)$ is a matching from \mathbf{N} to \mathbf{N} , i.e., from component types to sleeves. (Not all the sets $F(k)$ need be distinct matchings.) Here $|F|$ is a measure of the size of the feasible set, specifically the number of distinct

matchings among the indexed sets $\{F(k)\}_{k \in K}$. This in turn is the number of machine setups, and also of clusters, since a cluster consists of all products sharing an identical setup. The first term in (1) therefore represents setup time and the second processing time; together they constitute total manufacturing time.

It is important to realize that the problem does not require sequencing of board types, as the processing time of a subset is independent of the sequence of boards and the total processing time is independent of the sequencing of batches. Two special cases of ICMS are of special interest because they frequently arise in the fabs, and also can be solved efficiently.

2.2 Special Case: Single Common Setup

In this case, there is one setup common to all jobs and the problem reduces to that of finding a single setup to minimize processing time. We let $r_i = \sum_{k \in K} b^k r_i^k$, summing the requirements over all jobs for component i . An optimal solution can be obtained by re-indexing the components in nonincreasing order relative to the requirements $\{r_i\}$ and simultaneously indexing sleeve location j by nondecreasing pick and placement time $\{d_j\}$. This fact follows a classical result of Hardy, Littlewood and Polya (1952), which ensures that $f(\mathbf{r}, \mathbf{d}) = \sum_{i \in N} r_i d_i$ is minimized when $\{r_i\}$ and $\{d_i\}$ are arranged monotonically in opposite senses. The greater the requirement of a component type, the closer its assigned sleeve should be to the home position, see Figure 2.

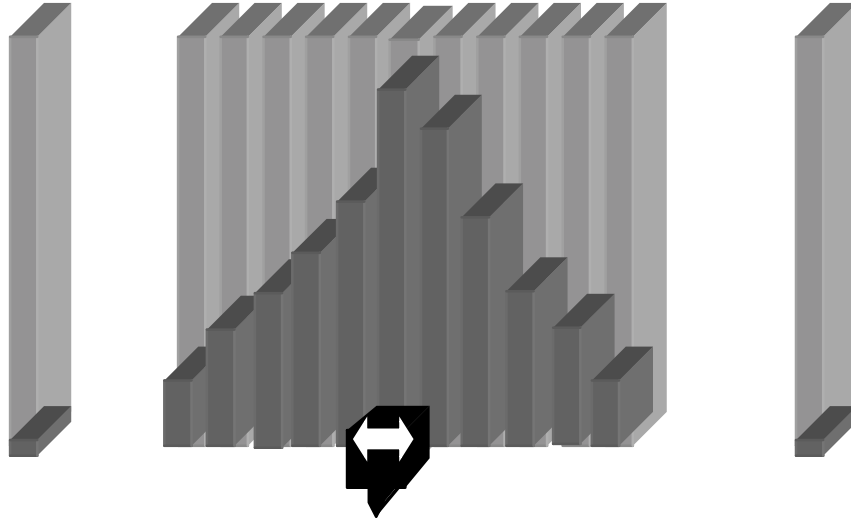


Figure 2. An illustration of an optimal assignment of components to sleeves. The greater the requirement of a component type, the closer its assigned sleeve to the home position.

This “organ pipe” arrangement familiar in material handling is relevant even to surface mount pick-and-place machines, which can have multiple insertion heads and dimensions of movement:

The Fuji engineers showed them that if they put the component feeders of the most heavily used parts near the center of the feeder bank, the insertion robot would not have to move as far to pick up the components. This reduction in travel time translated into processing time savings.

Here Johnson (1995) refers to the Fuji surface mount machines used at Hewlett-Packard.

Polak (forthcoming) further discusses conditions under which the organ pipe arrangement is an optimal machine setup.

2.3 Special Case: Fixed Sequence of Jobs

Our experience indicates that job sequences are predetermined, possibly by a simple dispatching rule such as first-come, first-serve, prior to the insertion step in the

assembly process. In this case, the jobs are processed in the fixed sequence $1, 2, \dots, K$.

Thus an admissible partition of the set \mathbf{K} must consist of consecutively indexed jobs. We can pose this as a shortest path problem on a network by defining a graph in which there is a node for each job k , as well as a dummy terminal node $(K+1)$. There is an arc (k, m) if node m follows node k in the given processing sequence and a setup is performed at job k for jobs $k, k+1, \dots, m-1$; it indicates an aggregation of jobs k through $(m-1)$. There is as well an arc $(k, K+1)$ for every node $k \in \mathbf{K}$, indicating a batch of jobs $k, k+1, \dots, K$. Each arc

has length $\lambda^{km} = \sigma + f\left(\sum_{q=k}^{m-1} r^q\right)$, that is, the sum of the setup and the optimal single setup

objective for an aggregation of jobs k through $(m-1)$. Given an admissible partition of the set of jobs \mathbf{K} into subsets, it is clear that the objective value is the length of a path from node 1 to node $(K+1)$ along the arcs corresponding to these subsets. Consequentially, a shortest path from node 1 to the terminal node $(K+1)$ gives an optimal solution to Model 2.

We note that Magazine and Polak (2002) consider this special case in the context of a constraint programming approach to ICMS.

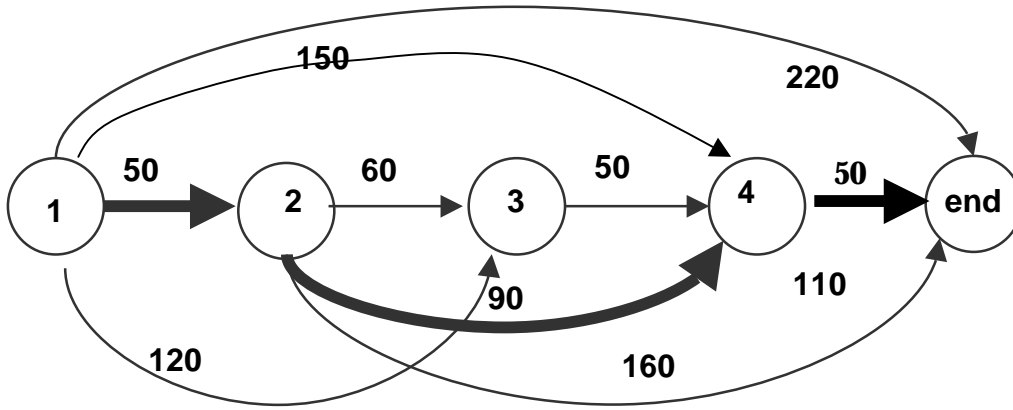


Figure 3. An example of the Network Model for the case of a fixed sequence of jobs. Boldface path indicates a setup for job 1, a common setup for jobs 2 & 3, and a distinct setup for job 4.

3. An Integer Programming Model

3.1 Formulation

For the general case in which there are no restrictions on the partition of the set of jobs \mathbf{K} , an integer programming equivalent to (1) is

$$\mathbf{IP} \quad \text{Minimize } z = \sigma s + \sum_{k \in K} \sum_{j \in N} \sum_{i \in N} c_{ij}^k x_{ij}^k \quad (2)$$

subject to

$$\sum_{i \in N} x_{ij}^k = 1, \forall j \in N, k \in K; \quad (3)$$

$$\sum_{j \in N} x_{ij}^k = 1, \forall i \in N, k \in K; \quad (4)$$

$$s = \text{number of distinct } (i,j)\text{-assignments in } \{x_{ij}^k\} \quad (5)$$

$$s \in \mathbf{Z}^+, x_{ij}^k \in \{0,1\}, \forall i, j \in N, k \in K. \quad (6)$$

Recall that $c_{ij}^k = b^k d_j r_i^k$; it indicates the time needed to insert all units of component i in batch k from sleeve j , and

$$x_{ij}^k = \begin{cases} 1, & \text{if component } i \text{ is loaded in sleeve } j \text{ for job } k, \\ 0, & \text{otherwise, } \forall i, j \in N, k \in K. \end{cases}$$

The objective (2) represents the total manufacturing time. Systems (3) and (4) ensure that there is a valid setup, i.e., an assignment of component types to sleeves, for each job k .

The number of distinct assignments (5) is exactly the number of setups required. To count these setups, we introduce a triangular array of binary variables $\{w_{km}: k < m, k, m \in \mathbf{K}\}$, where

$$w^{km} \geq x_{ij}^k - x_{ij}^m, \quad \forall i, j \in N, k, m \in K, m > k, \quad (7)$$

$$w^{km} \in \{0, 1\}, \quad \forall k, m \in K, m > k. \quad (8)$$

Thus if jobs k and m are processed together, i.e. following a common setup, w^{km} can be either 0 or 1 because $x_{ij}^k = x_{ij}^m$, for each $i, j \in N$. However, if jobs k and m are not processed together then we must have $w^{km} = 1$.

Next, for each job k , we introduce a variable $s^k \in \{0, 1\}, \forall k \in \mathbf{K}$ with the desired property that $s^k = 1$ if job k shares a common setup with any job m of higher index, and is 0 otherwise. Then the number of setups would be given by

$$s = K - \sum_{k \in K} s^k. \quad (9)$$

Toward this end we require

$$s^k \leq \begin{cases} \sum_{m>k} (1 - w^{km}), k < |K|, \\ 0, k = |K|. \end{cases} \quad (10)$$

Now observe that the objective coefficient, σ , of s in (2) is positive, forcing $s^k = 1$, and in turn $w^{km} = 0$ for some $m > k$, whenever $x_{ij}^k = x_{ij}^m$, for each $i, j \in \mathbf{N}$, in order to satisfy (10). This must hold in any integer solution that minimizes (2) subject to (3-4) and (6-10).

In light of the observations following (8), the feasible polytope of clustering and setup decisions is defined rather loosely, motivating the development of further valid equations and inequalities.

Analogous to s^k , we introduce a variable $v^m \in \{0,1\}$, $\forall m \in K$ with the desired property that indicates $v^m = 1$ if job m shares a common setup with any job k of lower index, and is 0 otherwise. The number of setups would then be

$$s = K - \sum_{m=1}^{K-1} v^m, \quad (13)$$

We require

$$v^m \leq \begin{cases} \sum_{k < m} (1 - w^{km}), \forall m > 1, \\ 0, m = 1, \end{cases} \quad (14)$$

Once again the fact that the objective coefficient, σ , of s in (2) is positive, forces $v^m = 1$, and in turn $w^{km} = 0$ for some $k < m$, whenever $x_{ij}^k = x_{ij}^m$, for each $i, j \in \mathbf{N}$, in any integer solution that minimizes (2) subject to (3-4), (6-8), and (13-14).

By analogy to (11) we have

$$v^m \geq 1 - w^{km}, \quad \forall k, m \in K, m > k, \quad (15)$$

Moreover as (13) sums across each row of the array w ,

$$\sum_{k=1}^{m-1} w^{km} = m - 1 - v^k, \quad \forall m \in \{2, \dots, K\} \quad (16)$$

sums down each column.

Initial upper bounds on the optimal value of **IP** are readily available. For the case of one common setup, $s = 1$, and for the case in which each job has its own setup, $s = K$, the objective (2) is easily computed as an upper bound. Solution of the shortest path model proposed in section 3.2 for any arbitrary sequence of jobs also provides an upper

bound. On the other hand, the optimal objective value of the LP relaxation to **IP** yields a lower bound on **IP**.

3.3 A Totally Unimodular Submatrix

Except for the single integer variable s , all variables in the formulation of **IP** are binary. Accounting for the latter, there are $[K(K-1)/2 - K]$ in the array w , KN^2 in the array x , and K in the superscripted s array. This is a substantial number for a problem of even moderate dimensions. Fortunately we can exploit the structure of **IP** to relax the binary requirements on the variables in x , the largest array, and still be assured that they take on binary values whenever w is binary. For example, with $N=16$ and $K=8$, there are 2076 binary variables in the original formulation, but only 28 in the relaxed formulation.

The key observation is that for fixed binary w , the submatrix of constraint columns in **IP** is totally unimodular. For it is well known that the assignment constraints (3-4) are totally unimodular; only in (7) does x otherwise appear. But it follows from Proposition 2.1 in Nemhauser and Wolsey (1988) that any difference of totally unimodular matrices, as in (7), is totally unimodular.

4. Numerical Study

4.1 Test Problems

Three series of moderately sized test problems afford a measure of insight into the computational requirements and behavior of model **IP**. We employed the AMPL language (Fourer, Gay and Kernighan, 1993) to generate these test problems. These in turn were processed by the CPLEX 6.0 Mixed Integer Programming (MIP) branch and bound routines implemented on a Pentium II processor at 400MH. In all instances,

- the pick and placement time $d_l = 20j$ for each sleeve j , where the sleeves are indexed from 1 , closest to the home position of the insertion head, through N , the farthest away.
- using an AMPL random generator, each component requirement was sampled from a uniform distribution from 5 to 20,
- using an AMPL random generator, each batch of PCB was sampled from a uniform distribution from 10 to 50.
- all valid cuts (11-16) were included in order to strengthen bounds and improve convergence.
- identical CPLEX MIP options were invoked, as detailed in the AMPL scripts displayed in Appendix 2.

The power and wide availability of the AMPL language and CPLEX solver should enable researchers and practitioners to readily reproduce and extend this numerical study.

The first series consisted of ten randomly generated instances of **IP** for the number N of components and sleeves fixed at 16 and the number K of board types at 8. There are 9602 constraints in 2064 continuous variables, 28 binary variables, and 1 integer variable in the formulation of **IP** with these dimensions. The AMPL presolve function eliminated 2091 constraints and the single integer variable, while the CPLEX MIP presolve function eliminated a further 16 constraints. Each instance was solved to optimality for 15 values of the unit setup time σ ranging from 20000 through 200000. Statistical means over the ten test problems are presented Table 1, and results of the first test problem alone in Table 2. As expected, the number of setups in the optimal solution comprise a nonincreasing sequence with respect to σ . The reported CPLEX CPU times

(in seconds) exhibit a general pattern over most of the problems: a broken climb as σ increases until s reaches the value 3 or 4, and a gradual decline beyond that point.

Table 1. Mean statistics for 10 randomly generated examples of IP, solved to optimality for each of 15 values of σ , the unit setup.

σ	Optimal No. Setups	CPU time	Nodes in B&B	No. Simplex Iterations	Optimal Objective
20000	8	115.4	200.6	16328.7	6238218
30000	7.4	180.1	449.1	32648.4	6315672
40000	6.7	266	722.1	54564.1	6384852
50000	5.8	327	930.3	70648	6446530
60000	5.3	420	1119.8	90071.8	6501448
70000	5	530	1320.4	112905.8	6553098
80000	4.5	583	1463.8	125333.8	6599644
90000	4.3	639	1538.2	140405.1	6644038
100000	3.8	669	1515	141198.4	6683978
110000	3.5	681	1529.7	146350.7	6720718
120000	3.2	701	1513.6	150658.5	6754426
130000	3.1	662	1343	137847.2	6785640
140000	2.9	739	1317.6	150515.1	6815662
150000	2.7	646	1155.8	128896	6844360
200000	2	647	892.1	113072.8	6966758

Table 2. Results of Problem 1 from the randomly generated set.

σ	Optimal No. Setups	CPU time	Nodes in B&B	No. simplex iterations	Optimal Objective
20000	8	60	163	14036	6466340
30000	8	81	299	21813	6546340
40000	6	110	404	31214	6610060
50000	6	140	561	46581	6670060
60000	6	170	708	53767	6730060
70000	6	340	990	79678	6790060
80000	5	390	1161	94299	6844500
90000	5	460	1312	103979	6894500
100000	4	500	1317	119669	6935420
110000	4	500	1312	116360	6975420
120000	4	580	1436	135988	7015420
130000	3	510	1261	112040	7047560
140000	3	440	1094	113926	7077560
150000	3	290	957	94777	7107560
200000	2	330	731	82495	7253140

The second series of randomly generated problems is based on problem 1 of the first series. The other problems in this series are generated by incrementing the number K of board types from 5 through 17 while holding the number of components and sleeves N at 16. We imposed a limit of 10,000 sec of CPU time on the CPLEX run for each instance. The results appear in Table 3. The instances for $K \leq 9$ solved to optimality within the time limit, while the relative gap = (best integer objective – best LP objective)/(best LP objective) in percent is displayed for the others.

Table 3. Numerical results for 12 instances based on Random Problem 1. The number N of components and sleeves remains constant at 16, while the number of board types K varies from 5 through 17. Here s is the number of setups in the best integer solution returned within the CPU time limit of 10,000 sec.

CPU time limit 1.0e+4 sec							
= 80000							
K	s	CPU time	B&B Nodes	No. iterations	Best Integer	Best LP	pct gap
5	3	17	76	8062	3287640	3287640	(optimal)
6	4	45	177	18497	4305220	4305220	(optimal)
7	4	250	762	85588	5453260	5453260	(optimal)
8	5	260	1161	94299	6844500	6844500	(optimal)
9	6	1100	3304	243723	8356500	8356500	(optimal)
10	6	Limit	8028	1254344	9376320	9245120	1.4%
11	6	Limit	12733	1256915	10030940	9969817	0.6%
12	6	Limit	8179	905029	10504800	10342000	1.6%
13	7	Limit	4986	680560	11343220	10994000	3.2%
14	8	Limit	2658	473773	12550840	12043000	4.2%
15	12	Limit	1746	384954	13996380	13355000	4.8%
16	11	Limit	1533	347239	14464080	13680000	5.7%
17	11	Limit	1243	319140	15524400	14614000	6.2%

Problem 1 is also the starting point for our third series of randomly generated problems in which the number K of board types is held constant at 8, while the number N

of components and sleeves is incremented from 16 through 30. The results appear in

Table 4.

Table 4. Numerical results for 15 instances based on Random Problem 1. The number K of board types remains constant at 8, while the number N of components and sleeves varies from 16 through 32. The unit setup time is $\sigma = 80000$ throughout the set.

= 80000							
N	Optimal No. Setups	CPU time	Continuous variables	Constraints	B&B Nodes	No. iterations	Optimal Objective
16	5	260	2064	7511	1161	94299	6844500
17	5	280	2328	8451	992	95902	7608100
18	6	360	2608	9447	769	82500	8618340
19	6	290	2904	10499	494	59671	9420920
20	6	400	3216	11607	706	101010	10620340
21	6	450	3544	12771	550	70652	11567280
22	7	610	3888	13991	424	66649	12570660
23	7	540	4248	15267	366	59753	13590060
24	7	840	4624	16599	409	90735	14933080
25	7	830	5016	17987	388	83341	16103120
26	8	2400	5424	19431	728	194653	17030760
27	7	470	5848	20931	249	62148	18275820
28	8	640	6288	22487	241	65862	19542860
29	8	500	6744	24099	160	45353	21245320
30	8	970	7216	25767	166	58477	22363420
31	8	520	7704	27491	114	39804	24113020
32	8	740	8208	29271	117	52461	25558800

4.2 Conclusions of the Numerical Study

Some general patterns emerge from the test problems. First, for a fixed number N of sleeves and components, and a fixed number K of board types, CPU time to optimal solution increases with σ , the unit setup time, then levels off and slightly decrease as σ becomes very large. Second, for fixed N and a fixed limit on CPU time, the relative gap between the best integer solution and best LP bound in the branch and bound tree increases with $K > 10$. Third, for fixed K the CPU time to optimality is relatively

insensitive to changes in N . Significantly, this last pattern suggests that **IP** is relatively computationally tractable for low mix, high volume PCB production.

5. Future Directions: Group Technology, Cluster Analysis, and Algorithms

The broader context of group technology and product clustering might admit further applications of **IP**. Group technology (GT) takes advantage of similarity within groups of products or parts, with goals that include curtailing machine setups, reducing work-in-process inventory, and improving work center balance. Relevant to PCB manufacturing, Carmon, Maimon and Dar-El (1989) proposed a heuristic group set-up (GSU) method for a two-machine PCB assembly process, with an overall objective of increasing throughput. Davis and Selep (1990) described the implementation of a “greedy board” GT heuristic to organize PCB board types into jobs, with a primary objective of reducing total setup time. Luzzatto and Perona (1993) proposed a multi-criteria heuristic for grouping PCB jobs, which they tested in turn by a simulation model.

Cluster analysis embodies the principal mathematical tools employed in GT. The p -median model, widely used for optimal clustering, seeks a fixed number p groups of objects to minimize a measure of total distance or dissimilarity. Mulvey and Crowder (1979) presented and tested an exact algorithm for optimal clustering, and Kusiak (1985) applied the model to GT. Ben-Arieh and Chang (1993) modified the p -median model to treat p , the number of clusters, as a decision variable. In the context of manufacturing, the objective in each case can be interpreted to be a surrogate for some measure of processing time or time.

Because **IP** prescribes groups of PCB jobs by common machine setup, we say that it clusters jobs by *process commonality*. Moreover, **IP** differs from the p -median model

and its generalizations in two significant ways. First, the objective function (2) explicitly accounts for processing and setup times and does not use a surrogate metric. Second, objects are not assigned to cluster medians in **IP**, but rather components of each object (PCB) are assigned to specific locations (sleeves) in the machine. These factors makes **IP** a more precise representation of any operation in which each setup is an assignment of parts to locations, or workers to work centers. A job shop in which any re-deployment of workers to machines is cause for a general “time out” in the shop would be a case in point.

In the fabs, we observed feederbanks with 24 sleeves (N), a similar number of component types, and mixes including up to 100 distinct board types (K). Our numerical study evidences the computational difficulty of solving **IP** in such a high mix environment, motivating the development and numerical testing of suitable heuristics. For instance, heuristic aggregation of board types into families might be an effective way to approximate a high mix fab by a more tractable low mix. Also, the formulation of ICMS in this work may prove useful for providing lower bounds for use in conjunction with the branch-and-price model of Cohn, Magazine, and Polak (2000).

Further development of exact algorithms could yield great benefits as well. The setup counting constraints (7-10) are in some sense reminiscent of the subtour elimination constraints in the traveling salesman problem (TSP). As is also the case with TSP, there is a rich collection of additional known valid inequalities (11-16). Therefore a branch and cut approach, so successfully applied to TSP beginning with the work of Padberg and Rinaldi (1991), might be a promising avenue for solving **IP**.

In conclusion, we have presented an alternative and original formulation to the ICMS model. Moreover, having identified two special cases commonly observed in fabrication shops, we prescribed an efficient method to solve each to optimality. Although for the general case ICMS is NP-hard, numerical testing suggests its relative tractability in low mix production planning. The capabilities of commercially available software do not presently permit the optimal solution of realistic high mix programs, which can include hundreds of board types. Development and testing of heuristics and exact algorithms, along with the identification of similar problems in group technology and cluster analysis, could greatly extend its applicability.

References

Ball, M.O. and M.J. Magazine (1988), "Sequencing of Insertions in Printed Circuit Board Assembly," *Operations Research* 36(2):192-201.

Bard, J.F. (1988), "A Heuristic for Minimizing the Number of Tool Switches on a Flexible Machine," *IIE Transactions* 20(4):382-391

Bard, J.F., R.W. Clayton and T.A. Feo (1994), "Machine Setup and Component Placement in Printed Circuit Board assembly," *International Journal of Flexible Manufacturing Systems* 6:5-31.

Ben-Arieh, D. and P.T. Chang (1994), "An Extension to the p-Median Group Technology Problem," *Computers and Operations Research* 21(2):119-125.

Carmon, T.F., O.Z. Maimon, and E.M. Dar-El (1989), "Group set-up for printed circuit board assembly," *International Journal of Production Research*, 27(10):1795-1810.

Cohn, A.M., M.J. Magazine and G.G.Polak (2000), Integrating Printed Circuit Board Manufacturing Problems by Branch-and-Price. Massachusetts Institute of Technology Operations Research Center Report OR 00-349, Cambridge, MA.

Cornuejols, G., M.L. Fisher and G.L. Nemhauser (1977), Location of Bank Accounts to Maximize Float: An Analytic Study of Exact and Approximate Algorithms. *Management Science* 23(8):789:810.

Crama, Y., J. van de Klundert, and F. C.R. Spieksma (1999), Production Planning Problems in Printed Circuit Board Assembly, GEMME 9925, University of Liege, Liege, Belgium.

Davis, T. and E. Selep (1990), "Group Technology for High-Mix Printed Circuit Assembly," *IEEE International Electronic Manufacturing Technology Symposium [proceedings]*, 264-269.

Fourer, R., D.M. Gay and B.W. Kernighan (1993), *AMPL: A Modeling Language for Mathematical Programming*, Boyd and Fraser Publishing Company.

Garey, M.R. and D.S. Johnson (1979), *Computers and Intractability: A guide to the Theory of NP-Completeness*, W.H. Freeman and Company.

Günther, H.-O., M. Grunow and C. Schorling (1997), "Workload planning in small lot printed circuit board assembly," *OR Spektrum* 19:147-157.

Hardy, G.J., Littlewood and G. Polya (1952), *Inequalities*, Second edition, Oxford University Press.

Hashiba, S. and T.-C. Chang (1991) "PCB Assembly Reduction Using Group Technology," *Computers and Industrial Engineering* 21(1-4):453-457

Hillier, M.S. and M.L. Brandeau (1998) "Optimal Component Assignment and Board Grouping in Printed Circuit Board Manufacturing," *Operations Research* 46(5):675-689.

Jain, S., M.E. Johnson and F. Safai (1996), "Implementing Setup Optimization on the Shop Floor," *Operations Research* 43(6): 843:851.

Johnson, M. Eric (1995), "Hewlett-Packard: Spokane Surface Mount Center," *Proc. of Decision Sciences*, Boston, Nov., 139-141.

Karp, R.M. (1975), On the Computational Complexity of Combinatorial Problems, *Networks* 5:45-68.

Kusiak, A. (1985), "The part families problem in flexible manufacturing systems," *Annals of Operations Research*, 3:139-154.

Luzzatto, D. and M. Perona (1993), "Cell formation in PCB assembly based on production quantitative data," *European Journal of Operations Research*, 69:312-329.

Magazine, M. J. and Polak, G. G. (2002), Job Release Policy and Printed Circuit Board Assembly, *IIE Transactions*, to appear.

Magazine, M. J., Polak, G. G. and Sharma, D. (2002), A Multi-Exchange Neighborhood Search Heuristic for an Integrated Clustering and Machine Setup Model for PCB Manufacturing, *Journal of Electronics Manufacturing*, 11(2):107-119.

Mulvey, J.M. and H. P. Crowder (1979), “Cluster Analysis: An Application of Lagrangian Relaxation,” *Management Science* 25(4): 329-339.

Nemhauser, G.L. and L. A. Wolsey (1988), *Integer and Combinatorial Optimization*, John Wiley & Sons.

Norman, S.K. (2001), Heuristic Approaches to Batching Jobs in Printed Circuit Board Assembly, Ph.D. dissertation, QAOM Department, the University of Cincinnati.

Padberg, M. and G. Rinaldi (1991) “A Branch-and-Cut Algorithm for the Resolution of Large-scale Symmetric Traveling Salesman Problems,” *SIAM Review*, 33:60-100.

Papdimitriou, C.H. and K. Steiglitz (1982), *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, N.J.

Polak, G.G. (forthcoming), “A Special Case of the Quadratic Assignment Model with an Application to Storage-and-Retrieval Devices,” in *Proceedings of the XI-th Latin-Iberian American Congress of Operations Research (CLAIO)*, 27-31 October, 2002, University of Concepción, Chile.

Rosing, K.E. and C.S. ReVelle (1986), “Optimal Clustering,” *Environment and Planning A* 18:1463-1476.

Sadiq, M., T.L. Landers and G.D. Taylor (1993), “A heuristic algorithm for minimizing total production time for a sequence of jobs on a surface mount placement machine,” *International Journal of Production Research* 31(6):1327-1341.

Appendix. A Numerical Illustration

Four distinct types of PCB must be assembled from four types of components in our small-scale illustration with nominal data. Each setup has a time of 100 seconds, while the processing or travel time is 1 second per component dispensed from sleeve 1, 2 seconds from sleeve 2, 3 seconds from sleeve 3, and 4 seconds from sleeve 4. Table 3 gives the component profiles and batch sizes for each job.

Job	Batch size	Component 1	Component 2	Component 3	Component 4
1	20	5	4	12	2

2	40	3	10	3	10
3	30	10	5	3	3
4	20	4	3	5	4

Table 3. Batch size and component requirements for each of 4 types of PCB.

Recalling that the objective coefficient of each component-sleeve assignment variable is

$c_{ij}^k = b^k d_j r_i^k$, the product of appropriate parameters of batch size, unit processing time, and

board profile, model **IP** is then

$$\text{Minimize } z = 100s + 100x_{1,1}^1 + 200x_{1,2}^1 + 300x_{1,3}^1 + 400x_{1,4}^1 + 80x_{2,1}^1 + 160x_{2,2}^1 \\ + \dots + 1200x_{2,3}^2 + 1600x_{2,4}^2 + \dots + 80x_{4,1}^4 + 160x_{4,2}^4 + 240x_{4,3}^4 + 320x_{4,4}^4$$

subject to

$$\sum_{i=1}^4 x_{ij}^k = 1, \quad \forall j \in \{1,2,3,4\}, k \in \{1,2,3,4\};$$

$$\sum_{j=1}^4 x_{ij}^k = 1, \quad \forall i \in \{1,2,3,4\}, k \in \{1,2,3,4\};$$

$$w^{1,2} \geq x_{ij}^1 - x_{ij}^2, \quad \forall j \in \{1,2,3,4\}, j \in \{1,2,3,4\}$$

$$w^{1,3} \geq x_{ij}^1 - x_{ij}^3, \quad \forall i \in \{1,2,3,4\}, j \in \{1,2,3,4\}$$

$$w^{1,4} \geq x_{ij}^1 - x_{ij}^4, \quad \forall i \in \{1,2,3,4\}, j \in \{1,2,3,4\}$$

$$w^{2,3} \geq x_{ij}^2 - x_{ij}^3, \quad \forall i \in \{1,2,3,4\}, j \in \{1,2,3,4\}$$

$$w^{2,4} \geq x_{ij}^2 - x_{ij}^4, \quad \forall i \in \{1,2,3,4\}, j \in \{1,2,3,4\}$$

$$w^{3,4} \geq x_{ij}^3 - x_{ij}^4, \quad \forall i \in \{1,2,3,4\}, j \in \{1,2,3,4\}$$

$$s^1 \leq (1 - w^{1,2}) + (1 - w^{1,3}) + (1 - w^{1,4})$$

$$s^2 \leq (1 - w^{2,3}) + (1 - w^{2,4})$$

$$s^3 \leq (1 - w^{3,4})$$

$$s = 4 - s^1 - s^2 - s^3$$

$$0 \leq x_{ij}^k \leq 1, \quad \forall i, j \in N, k \in K;$$

$$0 \leq s^k \leq 1, \quad \forall i, j \in N, k \in K;$$

$$w^{km} \in \{0,1\}, \quad \forall (k,m) \in \{(k,m) : k < m, k,m \in K.\}$$

There are a total of 214 constraints and 75 variables, 6 of them binary, in this

formulation, which does not include the valid inequalities presented in Section 3.3. The

presence of a totally unimodular submatrix, as discussed in Section 3, allows us to relax

the integrality on the \mathbf{x} and \mathbf{s} variables. The upper bounds of Section 3.3 were computed

to be 5250 seconds for the case of a single setup, $s = 1$, and 6010 seconds for $s = K = 4$

setups. The lower bound given by the LP relaxation is 5070 seconds, while the optimal objective value of **IP** is 5100 seconds.

CPLEX 6.5 was used to determine an optimal solution to **IP**, shown in Table 4; the **w** array in particular is displayed in Table 5. This solution calls for 3 setups: jobs 1 and 4 are clustered to share a common setup (note the “0” in cell (1,4) of Table 5), while job 2 and job 3 each has a distinct setup. Optimal setups can easily be determined following the result of Hardy, Littlewood and Polya cited in Section 2.2. The optimal objective value, which expresses total manufacturing time comprised of setup and processing time, is 5170 seconds.

W	Board type 1	Board type 2	Board type 3	Board type 4
Board type 1	-	1	1	0
Board type 2	-	-	1	1
Board type 3	-	-	-	1
Board type 4	-	-	-	-

Table 5. The **w array from the optimal solution reported by CPLEX 6.5. The “0” in the cell (1,4) indicates that types 1 and 4 are clustered for a common setup.**

Adding all valid cuts (11-16) to the formulation above allowed CPLEX to find the same optimal solution with fewer branch and bound nodes, requiring less CPU time as well. A comparison of the CPLEX results from both formulations appears in Table 6; note that the LP relaxation of **IP** was strengthened to 5100 seconds with the cuts from 5070 seconds without.

Table 6. Summary of CPLEX 6.5 Results for the illustration of model **IP, with and without the cuts of Section 3.3.**

Results from CPLEX 6.5 for illustrated IP		
	IP (without cuts)	IP (with cuts)
Optimal Value of Root	5070	5100

Node LP relaxation		
Branch & bound nodes to optimality	5	2
Simplex iterations to optimality	91	85
CPU (Solve) time, sec.	.200	.096
Number of constraints	214	231
Total number of variables	75	79
Number of binary variables	6	6