OPERATIONS RESEARCH CENTER

Working Paper

An Optimal Algorithm for Integrating Printed Circuit Board Manufacturing Problems

by
A.M. Cohn
M.J. Magazine
G.G. Polak

OR 349-00 **September 2000**

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Integrating Printed Circuit Board Manufacturing Problems by Branch-and-Price

Amy Mainville Cohn, Operations Research Center, The Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, amycohn@mit.edu

Michael J. Magazine, Department of QAOM, University of Cincinnati, Cincinnati, Ohio 45221-0130, michael.magazine@uc.edu

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

Abstract

Printed circuit boards appear in a wide array of products and thus their production is crucial to the contemporary electronics industry. A global approach to planning the complex and multi-stage production process is currently intractable. Nonetheless, significant improvements can be made by integrating closely related elements within the planning process. We focus here on the integration of two key problems -- *product clustering* and *machine setup*. In the product clustering problem, board types with similar component requirements are clustered together for assembly under a common configuration of the *pick-and-place machine*. In the machine setup problem, an optimal configuration of the pick-and-place machine is found for each of these clusters. In practice and in the literature, the product clustering and machine setup problems are typically solved sequentially. By instead solving the two problems simultaneously, we are able to find an optimal tradeoff between processing and setup times. We present the *Integrated Clustering and Machine Setup model* as a set partitioning problem. We describe a *branch-and-price* algorithm for solving this exponentially large problem. We introduce a *rank-cluster-and-prune*, a method for solving the imbedded pricing problems by combinatorial search, and conclude with computational results.

1. Introduction

Appearing in a wide range of products including home appliances, automotive controls, and computer hardware, printed circuit boards (PCB) play a crucial role in the contemporary electronics industry. Global PCB production in 1998 was estimated at $35 billion (Crama, van de Klundert, and Spieksma (1999)) with roughly half of that production occurring in the United States and Japan.

PCB production planning is complex and multi-staged, involving many interrelated decisions, and consequently raises many interesting and challenging research problems. Ball and Magazine (1988) demonstrate this complexity in their description of six principal steps – production order release, kitting, prepping, component insertion, component soldering, and board inspection. Crama, van de Klundert, and Spieksma (1999) propose a PCB planning hierarchy that includes eight different optimization subproblems, with significant attention paid to their interdependence. Both of these papers serve to highlight the challenges associated with efficiently planning the production process. In fact, a global optimization approach is presently intractable.

Nonetheless, integrating key subproblems can yield significant improvements. For example, Altinkemer, Kazaz, Köksalan and Moskowitz (2000) considered the subproblems of allocating components to feeders and of sequencing the placement of these components on the PCB. They noted that solving these independently, even to optimality, might result in "extremely poor" overall performance. They therefore formulated a single model to solve these two subproblems simultaneously, presenting numerical evidence of the benefits of such an integrated approach. Crama et al (1999) cite numerous other works that propose a joint approach to these same two subproblems.

Based on our association with a major computer manufacturer, we have focused our efforts on integrating two other key subproblems -- product clustering and machine setup, designated SP1 and SP5, respectively, by Crama et al (1999). Product clusters are groups of similar PCB types that proceed in common through the stages of production order release, kitting, and prepping in the planning sequence described by Ball and Magazine (1988). Machine setup determines the location on the assembly machine of

components for insertion and soldering. In section 2 we describe these problems in detail and consider the interaction between them. In section 3 we present the Integrated Cluster and Machine Setup (ICMS) model and formulate this model as a set partitioning problem. For many applications, ICMS also simultaneously determines for each cluster the assignment of component types to the machine. In this way ICMS incorporates a special case of yet a third subproblem, which Crama et al (1999) designate SP3 and call "partition" of components. In section 4 we describe a branch-and-price algorithm for solving this computationally challenging, large-scale integer program. We also present an original combinatorial search method, *rank-cluster-and-prune*, for solving the pricing problems required by the branch-and-price framework. As we show in section 4, this method employs a sorting operation in order to avoid the need for a polyhedral representation of the constraints characterizing a machine setup. In section 5 we present computational results for a number of test problems. We offer conclusions and suggest areas for further research in section 6.

2. The Product Clustering and Machine Setup Problems

2.1 Problem Descriptions

Printed circuit boards are made up of an assortment of components – capacitors, resistors, diodes, microprocessors, etc. – that are mounted onto a *substrate*, or blank board. The assembly plant that motivated this study employs *through-hole* mounting technology by which a computerized numerically controlled (CNC) *pick-and-place machine* inserts wire leads from the components into pre-drilled holes in the substrate. Through-hole technology is common for PCB products intended for high voltage or high amperage applications. For many other applications, *surface mount technology* (SMT)

obviates the need for pre-drilled holes in the substrate via advanced soldering techniques. Jain, Johnson and Safai (1996) and Moyer and Gupta (1996) describe SMT and the types of machines used to implement it in some detail. Though the assumptions of our model reflect a through-hole production environment, they are reasonably relevant to many SMT shops as well.

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

Figure 1 depicts a simplified version of the pick-and-place machine in use at the plant. It contains a component magazine known as a *feederbank*, which is a linear array of sleeves or slots used to store the various components. Each sleeve contains components of a single type. Suppose that each delivery and insertion from sleeve $j \in \{1,\ldots,N\}$ 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. Clearly retrieval time for a component must increase along with the distance of its

sleeve from the home position. While the component is being retrieved, the board is re-positioned under the insertion head and thus does not add excess time to the process. These operations are all numerically controlled by a combination of hardware and software.

Given that the re-positioning of the board occurs while the components are being retrieved, and that the time to insert the components is fixed and independent of the pick-and-place machine setup, an optimal machine setup is one that assigns component types to sleeves so as to minimize the total time spent in retrieving components.

In our discussion of how to construct optimal machine setups, we first consider the special case where a manufacturer only produces a single PCB, which requires r_i units of component type i, for each $i \in \{1,\ldots,N\}$. We let $s(\cdot)$ denote the function which assigns a unique sleeve to each component type in a setup. It is easy to see that the total retrieval time is minimized when component types are assigned to sleeves in the pick-and-place machine such that the more frequently a component type is used, the closer it is positioned to the insertion head. Figure 2 shows this "pipe organ" assignment s^*, familiar in material handling, which minimizes total processing time, $f(s) = \sum_{i \in N} r_i d_{s(i)}$. This follows from a classical result of Hardy, Littlewood and Polya (1952) which ensures that $\sum_{j \in N} r_j d_j$ is minimized when nonnegative sequences $\{r_j\}$ and $\{d_j\}$ are arranged monotonically in opposite senses. Thus, in this special case, finding an optimal machine setup reduces to simply sorting the component types by the frequency with which they appear on the PCB.

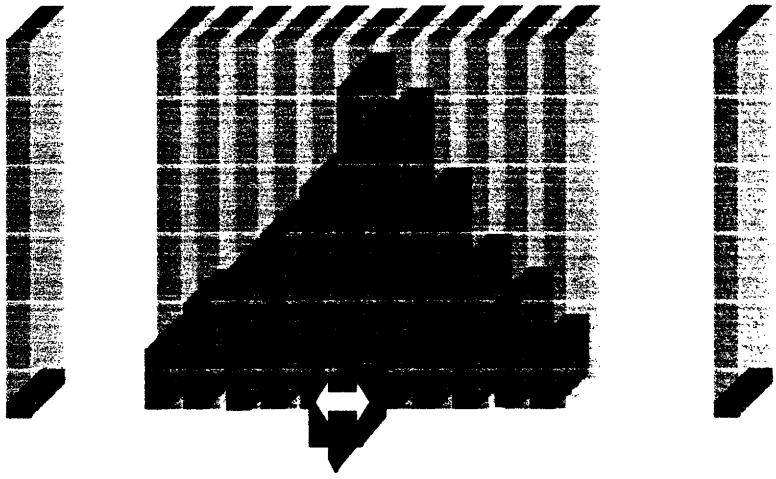

Figure 2. An optimal machine setup, i.e., assignment of component types to feeder sleeves. The greater the population of a type, the closer its assigned sleeve to the insertion head's home position.

We next extend the problem to the case where the manufacturer produces multiple copies of this single board type. It is clear that the machine configuration that is optimal for a single copy of the board type will also be optimal for the batch of multiple copies.

Finally, we consider the general case where the manufacturer produces a collection of distinct types of PCB, with varying quantities of each type. If the time associated with setting up the pick-and-place machine were negligible, then the optimal approach would be to re-configure the machine for each new board type, using the pipe organ approach described previously. (This had been the prior practice in the fabrication shop, or "fab," which we visited.) Conversely, if the time associated with setting up the pick-and-place machine were prohibitively high, then the optimal approach would be to manufacture all board types using one common machine configuration. To find an optimal configuration for this complete collection of board types, we would sum the component demands across *all* boards being manufactured, and then implement the pipe organ setup.

In reality, the machine setup time is usually somewhere in between – some groups of board types are similar enough to be processed together under a common machine setup to avoid excess setup time, while other board types merit individual setups so as to maximize processing efficiency. Thus in the literature, the planning process is typically broken down into two distinct steps. First, in the *product clustering problem*, similar boards types are clustered together, based on some approximation of processing time or cost. Next, a *machine setup problem* is solved for each cluster, which determines the *true* processing time or cost of each cluster and thus of the system overall. This machine setup problem is solved simply by the pipe organ assignment described earlier. Clearly, these two problems are strongly interrelated, and a sequential approach can lead to sub-optimality.

2.2 Literature Review

There is an extensive literature on the optimization of PCB operations, including much work that is relevant to clustering, machine setup, or component partition. Hashiba and Chang (1991) formulated an integer program to minimize the number of setups over all sequences of jobs in a PCB assembly shop, for which 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 et al (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.

Carmon, Maimon and Dar-El (1989) proposed a heuristic group setup 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" group technology 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. Ben-Arieh and Chang (1994) modified the p-median clustering model to treat p, the number of clusters, as a decision variable. Their objective of minimizing the total measure of dissimilarity among the clusters can be interpreted to be a surrogate for some measure of processing cost or time in the context of manufacturing.

The sequential approaches to clustering and setup extant in the literature often result in suboptimality for the joint problem. Therefore in section 3 we introduce the Integrated Clustering and Machine Setup (ICMS) model, within which these subproblems are solved simultaneously. Certain issues associated with ICMS are considered in two separate works. Magazine, Polak and Sharma (2001) show that this model is NP-hard,

and in addition propose and test a heuristic employing a multi-exchange neighborhood search. Based on that analysis of the computational complexity of ICMS, Magazine and Polak (2001) propose an alternative formulation for ICMS suitable for optimal solution of very small problem instances by off-the-shelf software. They also show in that work that the integrated model can be solved efficiently as a shortest path problem for the special case in which the assembler fixes the sequence of jobs released upstream from the assembly shop floor. Moreover they proceeded to analyze the opportunity cost of the organizational barriers that often result in such procedures on the shop floor.

We conclude this section with definitions and assumptions used in our work.

2.3 Definitions and Assumptions

A *job* is the production of all boards of one type and a *cluster* is any set of jobs. The *total manufacturing time* for a set of jobs consists of the sum of *processing* and *setup* times. Processing time is defined to be the total time spent by the pick-and-place machine in component retrieval and insertion. Setup time is the time associated with determining a machine configuration, where a *setup* is defined to be an assignment of component types to feederbank sleeves.

As noted in Section 2.1, the assumptions listed here are based on our experiences in through-hole fabrication shops, and also hold reasonably well in the many SMT shops: (1) **The time to "pick and place" a component is nondecreasing in its distance from the home position of the insertion head and is independent of the component's placement on the substrate.** The substrate is small relative to the feeder bank so that positioning the board for placement is relatively quick, while the component retrieval is the bottleneck operation.

(2) **Setup times are independent of the partition of jobs into clusters.** As in many PCB assembly operations, all setups that we observed in the fabrication shops 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 teardown 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)).

(3) The **feederbank sleeves are uncapacitated.** For the machine under consideration, to reload a sleeve with the same type of component is a simple operation requiring very little down time. Other machines are equipped with bulk feeders appropriate for many types of components. Moreover, the full teardown mode of setup affords an opportunity to restock all sleeves so that component "stockouts" are a rare occurrence during processing (Johnson (1995)). Altinkemer et al (2000) similarly noted that "the capacity of each feeder is not restrictive."

(4) Each **job k** belongs to one and only one cluster and **cannot be split among several clusters**. This is a common policy that, e.g., allows management to track a job for the purposes of monitoring quality.

3. The Integrated Clustering and Machine Setup Model (ICMS)

By solving the product clustering and machine setup problems simultaneously, we address the fundamental tradeoff between processing time (minimized by using a different setup for each board type) and machine setup time (minimized by manufacturing all boards under a common configuration). Before presenting the Integrated Clustering and Machine Setup model, we introduce the following notation:

- $\boldsymbol{K}$ denotes the set of jobs, where each job $k \in \boldsymbol{K}$ corresponds to a distinct board type;
- $\boldsymbol{N}$ denotes the set of component types, and also the set of sleeves in the feederbank of the pick-and-place machine;
- $\boldsymbol{C}$ denotes the set of all nonempty subsets of $\boldsymbol{K}$ (also referred to as *clusters*); $|\boldsymbol{C}|=2^{|K|}-1$;
- b^k denotes the number of boards in job k;
- r_i^k denotes the number of components of type $i \in \boldsymbol{N}$ required for each board of type k. Thus, job k requires $b^k * r_i^k$ components of type i in total;
- σ denotes the setup time associated with a single configuration of the pick-and-place machine;
- $s_c^*(i)$ denotes the sleeve assignment of component i in an optimal pipe organ machine configuration for cluster $c \in \boldsymbol{C}$.
- Recall that d_j denotes the time required to retrieve a component from sleeve j and insert it into the substrate.

Thus, the optimal processing time for cluster c is $f(s_c^*) = \sum_{k \in c} \sum_{i \in N} d_{s_c^*(i)} b^k r_i^k$,

which we denote by f_c^*. The optimal manufacturing time associated with cluster $\boldsymbol{c}$ is therefore $\sigma + f_c^*$. Our goal in the Integrated Clustering and Machine Setup model is to divide the set of jobs $\boldsymbol{K}$ into clusters so as to minimize the total manufacturing time. We can formulate this as a simple *set partitioning problem*. Let x_c represent the binary decision variable indicating whether cluster c is included in the solution ($x_c = 1$) or not ($x_c = 0$). The objective coefficient for x_c is simply $\sigma + f_c^*$, which is determined off-line, based on the pipe organ assignment of components to sleeves.

We have one constraint for each board type k stating that we must choose exactly one cluster containing that board type. The model can thus be written as

(ICMS)

$$\min \sum_{c \in C} (\sigma + f_c^*) x_c \qquad (1)$$

subject to

$$\sum_{c \in C:\ k \in c} x_c = 1 \qquad \forall \textit{ board type } k \in K \qquad (2)$$

$$x_c \in \{0, 1\} \qquad \forall \textit{ cluster } c \in C .$$

This model uses exact, rather than approximate, processing and setup times in product clustering, thereby leading to an optimal solution. This comes at the price, however, of an exponential number of variables, each of which requires us to solve a corresponding machine setup problem. Furthermore, these decision variables are binary, increasing the difficulty of solving the model.

It is only for convenience that we use the same index set N for sleeves and for component types; we need not assume that the number of component types is identical to the number of sleeves in the feederbank. Dummy components or sleeves can be employed as necessary. If there are more sleeves than types of components, a dummy component indicates that a sleeve is left empty in the appropriate setup. If instead the number of component types is larger, then a component assigned to a dummy sleeve associated with a suitably large d_j is actually inserted offline. This can be done manually or by a flexible pick-and-place machine as a follow up operation, as described by Hillier and Brandeau (1998). Thus our model determines the set of components to be loaded onto the machine for each cluster. For the special case of an assembly process that

employs a single principal pick-and-place machine, this is subproblem SP3 that Crama et al (1999) call the "partition" of components.

In section 4, we describe a branch-and-price approach that allows us to solve this exponentially large problem without explicitly considering the full set of potential clusters. Computational results in section 5 show that this approach enables us to solve problem instances of realistic size.

4. Solving ICMS With Branch-and-Price

We have formulated ICMS as an integer program with an exponential number of variables. To solve this computationally challenging problem, we use *branch-and-price*, a technique that combines *branch-and-bound* with *column generation*. Branch-and-price can be a powerful tool for solving large-scale IP's. Its success, however, is dependent on the careful design of key elements of the algorithm. [See Barnhart, Johnson, Nemhauser, Savelsbergh, and Vance (1998) for a detailed analysis of branch-and-price, along with numerous examples of its application.] In the sections that follow, we review the general concept of branch-and-price and then discuss the ICMS implementation.

4.1 Review of Branch-and-Price

Integer programs are frequently solved using *branch-and-bound*, a technique in which a series of increasingly constrained linear programming relaxations are solved until a provably optimal integer solution is found (Nemhauser and Wolsey (1988)). In large problems such as ICMS, even solving the individual LP relaxations can be challenging, due to their exponential number of variables. *Column generation* (Wolsey (1998)) is a useful tool for solving such large LP's. The purpose of column generation is to solve an LP with a very large number of variables, or *columns*, without explicitly

considering them all. We begin by solving a variation of the LP, known as the *restricted master problem*, in which only a subset of the variables are considered. Given the dual values associated with the solution to the restricted master, we then solve a *pricing problem* -- an optimization problem that identifies the most negative reduced cost variable from the original LP. If this variable has strictly negative reduced cost, then it is a valid pivot variable. We add it to the restricted master, re-solve to generate new dual values, and repeat. When the optimal value to the pricing problem is zero, we have established that no negative reduced cost variables exist and therefore the solution to the restricted master problem is also optimal for the original LP.

The success of column generation depends on our ability to identify a pricing problem which is fast enough to be solved many times, in spite of its large number of feasible solutions (note that a variable in the original LP corresponds to a feasible solution in the pricing problem). The structure of this pricing problem varies from application to application; in section 4.2, we discuss the ICMS pricing problem.

When column generation is used with branch-and-bound in a branch-and-price framework for solving large-scale IP's, it is not enough to design a pricing problem that can be solved quickly. This pricing problem must also be coordinated with a *branching strategy* for the branch-and-bound tree. Recall that in branch-and-bound, we solve an LP relaxation of the problem. If the solution is integer, we stop. Otherwise, we create two new sub-problems by adding constraints to the original LP such that the current fractional solution is not valid for either sub-problem, but all feasible solutions to the IP remain feasible for at least one of the sub-problems. When we then solve the two new LP's using

column generation, we must be able to enforce these added constraints within the pricing problem.

4.2 Branch-and-Price for ICMS

4.2.1 The ICMS Pricing Problem

In *ICMS*, the reduced cost of a cluster of board types c can be written as

$$\sigma + \sum_{i \in c} (f_c^*(i) - \pi_i), \tag{3}$$

where $f_c^*(i)$ is the processing time for job i given an optimal machine configuration for cluster c, and π_i is the dual variable associated with the cover constraint (2) for board type i.

The ICMS pricing problem seeks a cluster c in order to minimize the reduced cost less setup time, that is:

$$\min \sum_{i \in c} (f_c^*(i) - \pi_i) \tag{4}$$

subject to

$$c \in C. \tag{5}$$

If the solution to this optimization problem has value strictly less than $-\sigma$, then we have identified a new pivot variable. Otherwise, we have established optimality.

There are a number of ways to approach this pricing problem. We initially attempted to model it as an integer program. Unfortunately, slow convergence characterized all of the IP pricing models that we considered. The cause, common to all of them, was the presence of an imbedded matching problem containing N^2 binary variables, in which x_{ij} represents the decision to place component type i in sleeve j. Although this allowed us to capture the exact time of PCB processing, it resulted in a

very large number of binary variables. Due to their impact on the objective function, fractional values for these variables were very common, resulting in a large branch-and-bound tree.

In practice, we found that these matching variables often created a computational bottleneck. In other words, the IP pricing problems were spending an enormous amount of time determining the configuration of the pick-and-place machine. This is particularly vexing, given that it is trivial to compute an optimal configuration and processing time for a cluster of board types – this is simply the pipe organ problem! Based on this observation, we decided to instead take a combinatorial approach to the pricing problem.

***Rank-Cluster-and-Prune (RCP)* – A Combinatorial Approach**

RCP is a combinatorial algorithm that leverages the simplicity of the pipe organ configuration in order to identify a column with the most negative reduced cost. By employing sorting, an easy computational task, RCP obviates the need for a polyhedral representation of component-to-sleeve matching. This algorithm is basically a smart enumeration of all possible clusters, and consists of three operations – *ranking* the board types, computing the *cluster* processing time, and *pruning* the tree.

Rank. As we build and evaluate clusters in our combinatorial tree, we want to identify clusters with negative reduced cost as quickly as possible. Therefore, we would like to begin by considering those board types that seem most likely to be part of a negative reduced cost cluster.

Recall from (**3**) that the reduced cost of cluster c is $\sigma + \sum_{i \in c} (f_c^*(i) - \pi_i)$. Based on the fact that the subtraction of the duals π_i is the only negative contribution to a cluster's

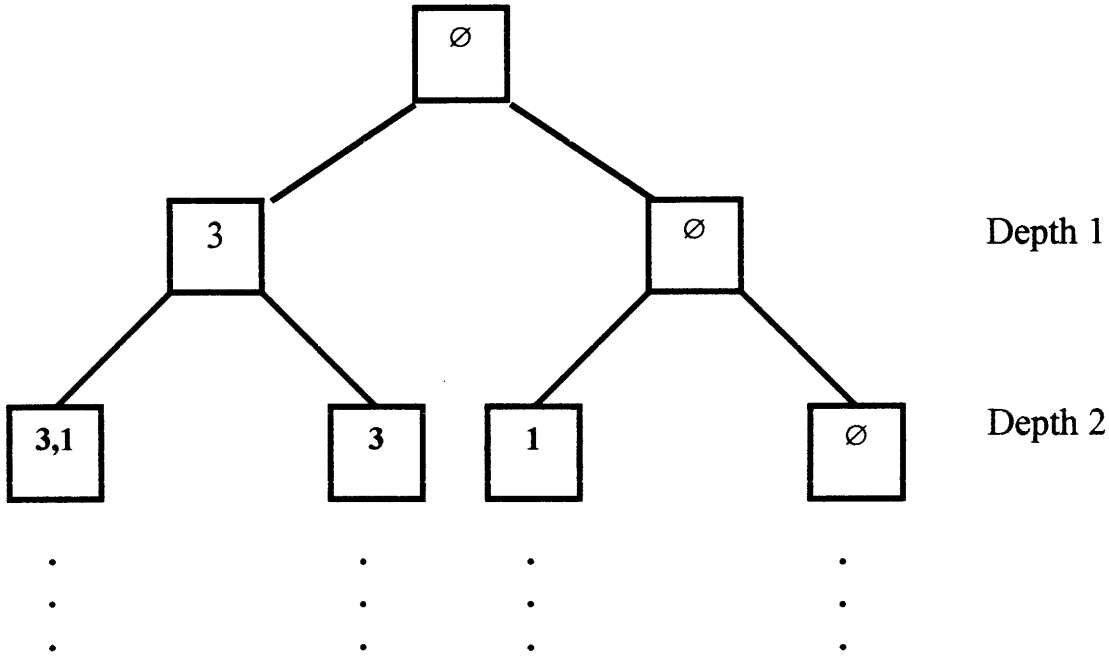

Figure 3. The Cluster Building Tree.

reduced cost (given that σ and f_c^* are necessarily positive), we rank the board types in decreasing order by dual value.

Cluster. Given this ranking, we then begin to construct a search tree in which each node represents a cluster of board types. The decision of whether to include the board type of rank *l* is made at depth *l* of the tree. The root node of this tree corresponds to the empty cluster, with reduced cost zero. We first consider whether to include the highest ranked board type. For the purpose of illustration, suppose that board type 3 is highest ranked, and board type 1 next highest. The cluster containing board type 3 has reduced cost $\sigma + f_3^*(3) - \pi_3$. At the next level we create two new clusters, {3,1} and {1}, with reduced costs $\sigma + f_{\{3,1\}}^*(3) - \pi_3 + f_{\{3,1\}}^*(1) - \pi_1$ and $\sigma + f_1^*(1) - \pi_1$. Figure 3 depicts this stage of the tree.

If we proceed in this fashion until the tree is fully constructed, then we can identify the most negative reduced cost cluster. Furthermore, given that the simplex method does not require the most negative reduced cost column, but simply a pivot variable with a reduced cost less that zero, we can abort our construction of the clustering tree as soon as we find *any* negative reduced cost cluster. Alternatively, we might choose to continue constructing the tree either until some specified number of negative reduced cost clusters has been found or until the tree has been completely enumerated. We can then add multiple columns to the restricted master. The simplex algorithm may in turn make several pivots, resulting in more accurate dual values for the next pricing problem.

The fact that we can generate multiple columns in one iteration of the pricing problem is a clear advantage over an IP approach. Furthermore, we leverage the simple pipe organ method for identifying optimal machine configurations. However, a full enumeration of the tree at each instance of the pricing problem would render the column generation approach intractable. In order to avoid computing the reduced cost of all $2^{|K|}$ clusters, we need to utilize a pruning operation.

Prune. Consider a node representing some cluster c with non-negative reduced cost, and denote the set of remaining board types yet to be considered in this part of the tree by $\boldsymbol{R}$. If we can prove that for any subset $r \subseteq \boldsymbol{R}$ the reduced cost of $c \cup r$ is non-negative, then we do not need to investigate this portion of the tree any further and we can prune this node.

How can we establish this fact? Consider the ways in which adding new board types to a given cluster can affect the cluster's reduced cost: (i) the processing time for the new board types is strictly non-negative and therefore increases the total processing

time; (ii) the new optimal configuration of the pick-and-place machine may also increase the processing time for the original board types; (iii) the reduced cost decreases by the dual value of the additional boards. Based on these three facts, we first prune the tree by observing that we can disregard any board type j for which π_j is non-positive, because all three components of the reduced cost will increase.

A second way to prune the tree is based on the fact that when we add board type j to cluster c, we not only *subtract* its dual value from the reduced cost of the original cluster, but also must *add* its processing time (as well as potentially increasing the processing time of the existing board types in c as well). Thus, if $f_j^*(j) - \pi_j \geq 0$,then the reduced cost of any set $c \cup j$ will always exceed the reduced cost of c, because the optimal processing time of job j is a lower bound on its processing time within cluster c. Therefore, we can disregard any board type j for which $f_j^*(j) - \pi_j \geq 0$.

Although these two approaches can decrease the problem size by eliminating some of the board types from consideration, in our computational experience the remaining tree still required excessive examination. To further prune the tree, we introduce the concept of *node potentials*. As noted above, when we add board type j to subset c, we modify the reduced cost of the cluster by at least $f_j^*(j) - \pi_j$. Given that we disregard all board types for which $f_j^*(j) - \pi_j \geq 0$, it is clear then that the following is true:

$$\sigma + \sum_{i \in c} (f_{c \cup r}^*(i) - \pi_i) + \sum_{i \in r} (f_{c \cup r}^*(i) - \pi_i) \geq$$
$$\sigma + \sum_{i \in c} (f_c^*(i) - \pi_i) + \sum_{i \in r} (f_i^*(i) - \pi_i) \geq$$
$$\sigma + \sum_{i \in c} (f_c^*(i) - \pi_i) + \sum_{i \in R} (f_i^*(i) - \pi_i).$$

Thus, the reduced cost of any cluster found on the branch stemming from cluster c can be bounded from below by $\sigma + \sum_{i \in c}(f_c^*(i) - \pi_i) + \sum_{i \in R}(f_i^*(i) - \pi_i)$.

If this lower bound is non-negative, then the node can be pruned.

From a computational standpoint, it is important to note that this bound does not depend on the given choice of cluster, but only on the depth of the tree, i.e. the set of board types remaining to be considered. At the beginning of the algorithm, we can compute node potentials in the following way. Let $p(j)$ denote the potential associated with depth j. Then $p(|K|) = f_{|K|}^*(|K|) - \pi_{|K|}$ and $p(j) = p(j+1) + f_j^*(j) - \pi_j$ for all others. Thus, the node potentials can be computed efficiently.

Additional, more complex pruning techniques were also used to enhance the performance of RCP. We direct the reader to Cohn and Polak (2000) for a more detailed discussion.

4.2.2 The ICMS Branching Strategy

To solve ICMS with branch-and-price, we also need to determine a branching strategy. An effective strategy in this case is to branch on pairs of board types. That is, we branch on dichotomies such as "board types k_s and k_t {are/are not} in the same cluster." This is the branching rule developed by Ryan and Foster (1981); see Barnhart et al (1998), Wolsey (1998) and Mehrotra, Natraj and Trick (2001) for other applications of this strategy.

We can easily incorporate these constraints in the pricing problem. First consider the branch in which k_s and k_t are in the same cluster. To enforce this, we simply replace board types k_s and k_t by a composite board type, denoted $k_{s,t}$, for which the total number of components of type i is $(b^s \cdot r_i^s) + (b^t \cdot r_i^t)$, update the restricted master accordingly,

and re-solve for the next set of dual values. Note that in doing so, we also decrease the depth of the pricing tree by one.

We next consider the branch in which k_s and k_t are *not* in the same cluster. It is also easy to ensure that at most one of these board types is included in the newly generated column. Assume without loss of generality that k_s is ranked higher than k_t, that is, k_s is considered before k_t in the combinatorial pricing tree. In branches of the tree for which k_s has been rejected, clearly the separation is enforced. In branches for which k_s has been accepted, we simply remove k_t from the list of pending board types yet to be considered.

5. Numerical Study

5.1 Problem Instances

We based our computational experiments on the test set devised by Norman (2000), considering seven production scenarios in all. For each scenario the number of board types $|K|$ was chosen from the set {10,32,60} and the number of components $|N|$ from {16,100}. Additionally, there are three possible component profiles: a *uniform* profile indicates that all component requirements were sampled from a single uniform distribution; a *60/40* profile indicates that 60% of the component requirements were sampled from one uniform distribution and 40% from another; and an *80/20* profile indicates that 80% of the component requirements were sampled from one uniform distribution and 20% from another. In all cases the number of machine sleeves is identical to the number of component types, and the time, measured in nominal units we call seconds, required to retrieve a component from sleeve j and insert it into the substrate

is simply $d_j = j$ seconds. (These are features of the test set, neither of which is required by our model.)

For each of these seven scenarios, we looked at a wide range of setup times in order to assess the impact of σ on the model's performance. We began by arbitrarily setting σ to 10,000 seconds. After solving this problem instance, we then repeatedly divided σ by two and re-solved, until the setup time became small enough that an optimal solution was to process each board type under its own optimal machine configuration. Similarly, we repeatedly doubled σ and re-solved until the setup time was large enough that an optimal solution was to cluster all boards together for a single machine setup. Computational results for these problem instances are provided in section 5.3.

5.2 Implementation Issues

We implemented the branch-and-price algorithm in C++ with CPLEX 7.0. It was run on a desktop PC with a Pentium III processor running at 800 MHz with 384 megabytes of RAM.

For the initial restricted master problems, we included columns representing all clusters of size 1, 2, 3, $|K|$, $|K|$-*1*, $|K|$-*2*, and $|K|$-*3*. We used CPLEX's dual simplex function to solve the LP's and in the pricing problem, we attempted to generate multiple columns. This parameter was normally set at either 100 or 500 columns with negative reduced cost per iteration, along with a number of neighboring columns. However, given that the number of available negative reduced cost columns decreases as the quality of the dual values improves, we also limited the size of the pricing tree. Typically, we would not allow the tree to exceed 2×10^6 nodes unless no new columns had yet been identified;

in this case, we terminated the pricing problem as soon as the first new column was found.

For those problem instances in which the root node had a fractional solution, we implemented branch-and-bound. To create the branching dichotomy discussed in Section 4.2.2 required several steps. First, the 0-1 matrix comprised of the fractional columns of the restricted master problem was arranged in totally reverse lexicographical (TLR) order using a polynomial algorithm described in Nemhauser and Wolsey (1988). Within this TRL matrix, we then identified an instance of the so-called "F-matrix" $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ as a submatrix, the row indices of which correspond to two board types we denote s and t. On one branch, the pricing problem is modified to require boards s and t to belong to the same cluster; on the other branch, to belong to different clusters.

5.3 Computational Results

Table 1 displays results for the root node (i.e. the initial LP relaxation) for instances of the seven planning scenarios. The first two columns display the number of board types and number of components for each scenario. The next column displays the setup time; note that multiple setup times are considered for each scenario. This is followed by the run time, in seconds. For those instances in which the solution to the LP relaxation was integral, we next display the number of clusters in an optimal solution. Otherwise, **F*** is used to indicate a fractional solution. We next indicate how many times the pricing problem was called. Finally, we indicate how many of those calls occurred after an optimal solution was found, i.e. how many pricing problems were required to prove optimality.

Table 1. Root node reults for instances of ICMS in seven PCB planning scenarios. A cell marked by a double asterisk (**) indicates that the corresponding LP relaxation could not be solved within the time limit of 5×10^6 CPU seconds.

Num. Boards (type of profile)	Num. Comp.'s	Setup Time	CPUTime (seconds)	Optimal Number of Clusters	Number of Calls to Pricing Problem	Calls Beyond LP Optimal
Scenario 1		10,000	1	10	1	0
10	**16**	20,000	1	9	1	0
(60/40 profile)		40,000	1	F*	1	0
		80,000	1	3	3	1
		160,000	1	2	3	0
		320,000	1	1	2	1
Scenario 2		80,000	1	10	1	0
10	**100**	160,000	1	9	1	0
(uniform profile)		320,000	1	5	1	1
		640,000	1	2	2	0
		1,280,000	1	1	2	1
Scenario 3		40	1	32	1	0
32	**16**	79	1	31	1	0
(uniform profile)		157	1	30	1	0
		313	1	28	1	0
		625	1	23	1	0
		1,250	1	16	1	0
		2,500	2	F*	2	0
		5,000	11	F*	3	0
		10,000	92	F*	15	0
		20,000	1016	3	32	8
		40,000	4012	F*	59	0
		80,000	68262	1	204	203
Scenario 4		5,000	1	32	1	0
32	**16**	10,000	1	25	1	0
(60/40 profile)		20,000	1	17	2	1
		40,000	7	10	4	0
		80,000	76	F*	14	0
		160,000	2119	F*	51	0
		320,000	10229	F*	67	1
		640,000	2723	1	27	26