A Minimum Manual Component Insertion Algorithm for PCB Assembly Based on Graph Theory

Thomas Lacksonen and Sanjay Joshi, Pennsylvania State University, University Park, PA

Abstract

In automated electronic printed circuit board assembly, it is cheaper to automatically assemble as many components as possible. This is done by creating a process plan that minimizes the number of conflicts between a component to be inserted and previously inserted components. This paper describes an algorithm based on graph theory that alleviates these conflicts by minimizing the number of PCB components that must be inserted manually. The algorithm handles parts that can be gripped in two possible directions. It can be used as an improved potential approach to process planning for PCBs.

Keywords: *Printed Circuit Board Assembly, Process Planning, Graph Theory*

Introduction

The major trend in printed circuit board (PCB) assembly is toward automated insertion of components.[1] This equipment can take two basic forms—automatic insertion equipment or robots.

Automatic insertion equipment can quickly install many standard components.[2] These machines often have insertion heads that can handle only one type of component (axial, radial, DIP) and are placed in series. More advanced machines have multiple heads for inserting different component types. Typically, the head can only move in the y direction or is stationary. The PCB holder can move in either the x or y direction. The head or the holder may rotate 90 degrees. The assembly cost for various component insertion sequences depends on the type of equipment used.

Robots with multiple heads are more flexible because they can insert any type of component in any direction. However, robots are slower because their heads must move between the component pickup point and the board. Robots are also limited in the number of components they can pick up at one station and are not as common as automatic insertion equipment in today's plants.[3]

The objective of both types of equipment is to minimize total assembly cost, which primarily involves reducing cycle time. Currently, on paper or electronically, a PCB layout is passed from the designer to the process planner (as shown in *Figure 1*). The designer can create a layout with no interferences by spacing the components farther apart than the insertion heads. This layout conflicts with the design objective of reducing board size. However, it is not possible for the designer to know which interferences will require manual insertion.

The process planner minimizes several cost factors. For PCB assembly, the operation planning summary must include a list of components to be assembled at each machine along with the sequence of assembly. Assembling all like components consecutively reduces the required number of mounting head changes, thus reducing the time. Sequencing based on travel distances, x and y direction speeds, and the number of rotations can reduce total assembly time. Reducing interferences minimizes the amount of manual assembly. Finally, when the sequence is developed based on these factors, it may be fed back to the designer as input for changes.

This paper presents an algorithm that minimizes the number of components which must be manually inserted due to interferences between the insertion head and previously inserted components. This is the first step of a two-step approach to process planning (shown by the dashed lines in *Figure 1*). This algorithm is proposed as an alternative to some of the single step and improvement approaches discussed in the next section.

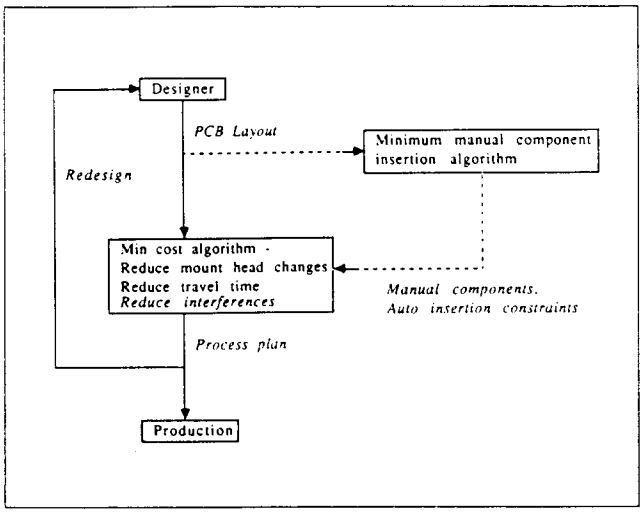


Figure 1
Flowchart for PCB Process Planning Current (solid) vs. Proposed (dashed)

Literature Review

With all of these potential constraints, developing the minimum cost process plan is a very difficult task. In a conventional system, "the previous experience of the process planner is critical to the success of the plan. Planning is as much an art as it is a formal procedure."[4] This manual process can be inaccurate and time consuming.

The planning problem has multiple objectives and a combinatorial structure. There has been limited research in modeling and optimizing the process. Two papers focus on sequencing PCB jobs. Randhawa et al[5] use integer programming to determine the number of sequencers, or component holders per machine. They assume that all like components are assembled consecutively. Cunningham and Browne[6] use artificial intelligence to sequence jobs on one automatic insertion machine to minimize setup costs.

Three rule-based approaches to process planning have been created. Bao[7] uses an expert system to assess the manufacturability of a PCB design. Some considerations are board layout, component spacing, and orientation. Cavalloro and Cividati[8] describe INCA which tries to minimize manual insertions and total processing time. In their approach, they generate a feasible sequence and try to improve this sequence by switching one auto-inserted component with a group of manually-inserted components. Chang and Terwilliger[2,9] have developed a prototype program called PWA_Planner that automates the entire planning process. Its sequencing module uses the following ranked rules to generate 'good' process plans:

1. Sequence components in the direction of the mounting head's largest clearance area requirements.
2. Sequence components per mounting head in a path taking a minimum amount of time (using a heuristic).
3. Sequence mounting heads in order of descending clearance area required.
4. Minimize idle time of each mounting head.
5. Sequence assembly stations in order of descending clearance area required by its mounting head(s).

These rules generally produce a small number of manually-inserted components. Their objective is to minimize total assembly cost, and not to minimize manually-inserted components or any other single cost element.

Problem Statement

A major assembly cost factor is manual insertion, which is generally five to twenty times more expensive than automated insertion.[10] Components need to be inserted manually if:

1) their unusual shape cannot be handled by the insertion heads,
2) there is an interference between the insertion head and previously inserted components or
3) other physical or processing restrictions exist.

Therefore, one objective in reducing costs is to minimize the number of components with interferences.

To approximate when an interference condition exists, one can define each component as a rectangle in the x-y plane of a rectangular PC board. A component being inserted can be defined as a larger rectangle that includes the gripper of the insertion head on two opposite sides of the component. *Figure 2* shows two possible gripper configurations. If known, the actual gripper size replaces this approximation. If the rectangle of a gripped component overlaps the rectangle of another component already inserted, there is an interference (as shown

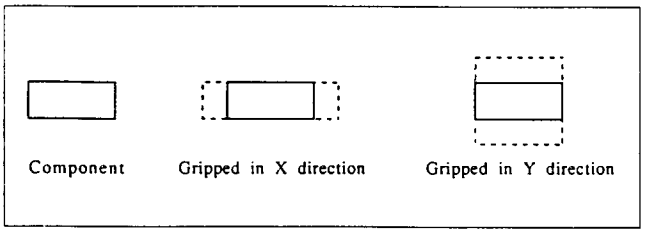


Figure 2
Gripper Configurations

in *Figure 3*). Notice that if the components are gripped in different directions, the interference becomes sequence dependent. Finally, some types of components can be gripped in either direction, creating another variable in determining when interference exists.

Each interference creates a precedence constraint which must be satisfied by the insertion sequence if both components are to be automatically inserted. As illustrated in *Figure 3*, component *A* must precede component *B* (or $A \rightarrow B$). If an assembly sequence maintaining all precedence constraints cannot be found, then some components cannot be inserted automatically. For example, no sequence can be found which satisfies $A \rightarrow B$, $B \rightarrow C$ and $C \rightarrow A$. The objective then is to remove the minimum number of components with their associated constraints to create a set of precedence constraints which can be maintained during automatic insertion.

It is possible for several different process plans to have the same number of manually-inserted components. Therefore, the algorithm created will only determine the manually-inserted components and the remaining maintainable set of precedence constraints. This information can be passed on to sequencing algorithms. These simplified but constrained algorithms will try to minimize costs by reducing travel time and mounting head changes. A secondary objective is to minimize the number of remaining precedence constraints, so that the algorithms which follow are constrained as little as possible. This proposed two-step approach to process planning is shown by the dashed lines in *Figure 1*.

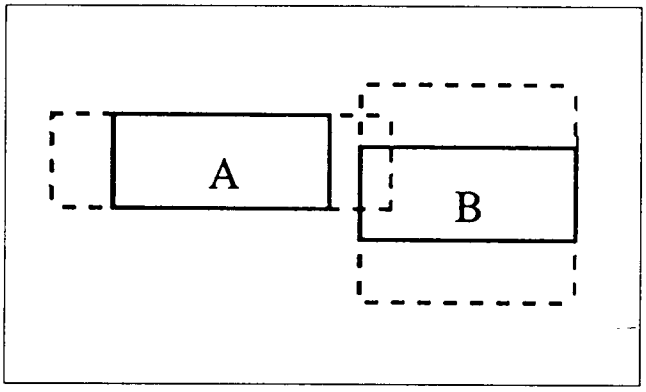


Figure 3
Component B Interferes with Insertion of Component A

Methodology

Integer Programming Formulation

The problem can be described by the following zero-one integer programming formulation.

$$Max \quad \sum_{i=1}^{n} \sum_{j=1}^{n} X_{ij}$$

$$S.\,T. \quad \sum_{j=1}^{n} X_{ij} \leq 1 \qquad i = 1, 2, \ldots n \qquad (1)$$

$$\sum_{i=1}^{n} X_{ij} \leq 1 \qquad j = 1, 2, \ldots n \qquad (2)$$

$$X_{kj} + X_{i1} \leq 1 \qquad (i,k) \in P \quad j = 1, \ldots n-1 \quad i = j, \ldots n \qquad (3)$$

$$X_{ij} = 0, 1 \qquad i = 1, 2, \ldots n \quad j = 1, 2, \ldots n \qquad (4)$$

Where X_{ij} = 1 if component i is assembled in the jth position in the sequence
0 otherwise
i = number of component
j = position in assembly sequence

Constraint (1) ensures that each component is assigned to the sequence only once.

Constraint (2) ensures that each position in the assembly sequence is assigned only one component.

Constraint (3) ensures that the precedence constraints between components (i,k) are not violated.

P = Set of precedence relations between components

$(i,k) \in P$—Component i must be assembled before component k.

A problem with N components and P precedence relations will require N^2 variables and $2N + PN(N-1)/2$ constraints. This will exceed the solution capabilities of computer solution packages for any reasonably-sized problem. Therefore, an alternative approach utilizing graphs will be used.

Graph Theory Representation

The precedence relationships can be represented by a directed graph $G = (V,E)$, where V = {set of nodes} and E = {set of directed arcs} such that for each component C_i {PCB assembly} there exists a node n_i in V, and each precedence constraint is represented as a directed arc a_{ij} $\{E\}$, where a_{ij} implies insertion of component C_i before component C_j. For a digraph, a cycle is defined as a closed (starts and ends at the same node) node-to-node walk along n distinct arcs.[11] A cycle represents a set of precedence constraints which cannot be maintained by any insertion sequence of components. A graph with no cycles is called an acyclic graph. The objective for PCB assembly is to find the minimum number of nodes, along with all attached arcs to be removed from the graph, to create an acyclic subgraph. These nodes represent the components which must be inserted manually. This problem is equivalent to finding the 'feedback vertex set' which is an NP-complete problem.[12,13] Therefore, this paper focuses on a heuristic solution.

The following definitions lead to the algorithm. The number of arcs going into a node (predecessors) is called the indegree, or d^+. The number of arcs going out of a node (successors) is called the outdegree, or d^-. Five types of nodes are defined based on indegree and outdegree.

1. A transmitter has $d^+ = 0$ and $d^- > 0$.
2. A receiver has $d^- = 0$ and $d^+ > 0$.
3. An isolate has $d^+ = 0$ and $d^- = 0$.
4. A carrier has $d^+ = 1$ and $d^- = 1$.
5. An ordinary is none of the above.

In sequencing, transmitters go first, receivers go last, and isolates go anywhere. Carriers do not impact the number of cycles in a graph. A key theorem in graph theory states that an acyclic diagraph has at least one transmitter node and one receiver node.[14] To create an acyclic diagraph, remove nodes which:

1) create receivers,
2) create transmitters, or
3) remove as many arcs as possible.

Removing arcs will eventually lead to the creation of receivers and transmitters. These goals are the basis of the algorithm.

The graph representation must be extended to handle components that can be assembled by gripping in either direction. Two nodes are defined for each of these components. Node A is for horizontal gripping and node A' is for vertical gripping. These two nodes have the same predecessor but different successor constraints. However, one of these nodes should be automatically inserted. To ensure the 'manual insertion' of one of the nodes, two dummy precedence constraints are also added: $A \Rightarrow A'$ and $A' \Rightarrow A$. Finally, the output of the algorithm must convert this data back into terms of the original components and list the direction in which these components are to be inserted.

Algorithm

The minimum component algorithm is described below and is summarized in *Figure 4*.

Step 0—Initialization. Read PCB layout data and calculate precedence constraints.

All of the precedence constraints and components start as unscheduled—to be defined later as either manually or automatically inserted. First, each component's coordinates and orientation are read in based on the PCB design. Then each component plus gripper rectangle is compared to the rectangle locations of all other components. Each overlap of rectangles identifies a precedence constraint. Then dummy nodes and constraints are added for two-directional components.

Step 1—Auto insertion components. Transmitter, receiver, and isolate components are scheduled for automatic insertion.

The indegree and outdegree of each unscheduled component is calculated based only on unscheduled constraints. Each component with $d^+ = 0$ (transmitter), $d^- = 0$ (receiver) or $d^+ + d^- = 0$ (isolate) is scheduled for automatic insertion. Also, all constraints with these components as a predecessor or successor are scheduled for automatic insertion. If all components have been scheduled, go to step 3. If some components were scheduled, repeat step 1 with a shorter constraint list. If no components were scheduled, go to step 2.

Step 2—Manual insertion components. The component which creates the most transmitters, receivers, and isolates is scheduled for manual insertion.

All components with $d^+ = 1$ or $d^- = 1$ are analyzed. Based on graph theory, these can become transmitters or receivers for auto insertion by the scheduling of only one component for manual insertion. For each component, the endnode value E is found, where E is the number of components for which the component is the only predecessor or successor. The component with the maximum E is scheduled for manual insertion. Often, there is a tie for the maximum E which is broken by the component with the maximum $d^+ + d^-$. This tiebreaker should reduce the number of manually-inserted components needed and reduce the number of auto-inserted constraints. Adjustments need to be made when analyzing carrier nodes ($d^+ = 1$ and $d^- = 1$). If a carrier has the same predecessor and sucessor, double counting must be avoided. For example, if the only constraints for A are $A \rightarrow B$ and $B \rightarrow A$, then inserting B manually allows one component to be auto inserted, not two. For this example $E = 1$ for component B. If the only predecessor and successor are different, this should not be included in the calculation of E. For example, if the only constraints for B are $A \rightarrow B$ and $B \rightarrow C$, then it may not be necessary to insert either A or C manually. For this example, $E = 0$ for components A and C. Upon selecting the component for manual insertion, all constraints associated with the component (both unscheduled and auto inserted) are changed to manual constraints. Finally, since this step should create components for auto insertion, go back to step 1.

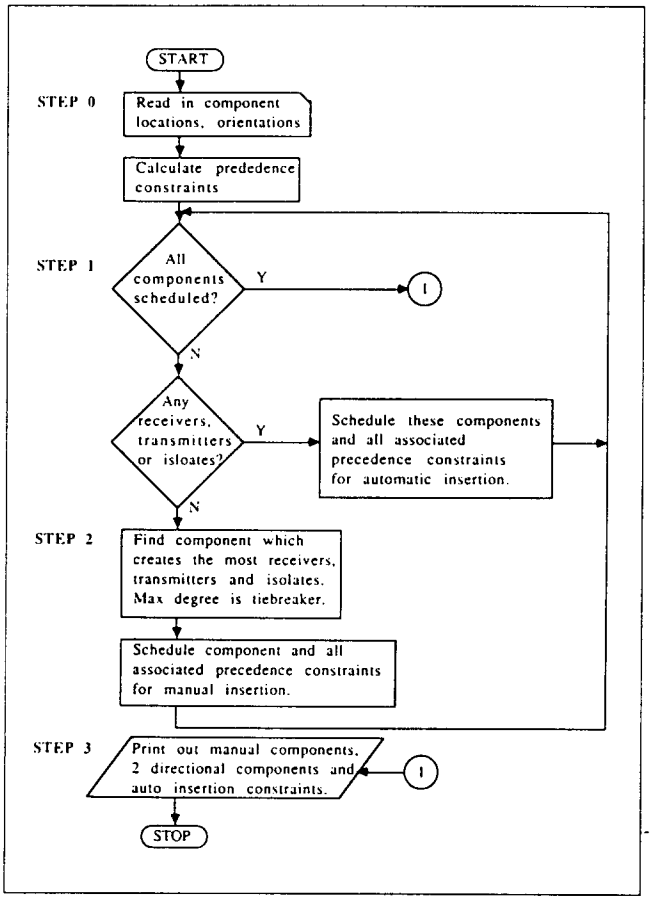


Figure 4
Flowchart of Algorithm

Step 3—Output of results.

When all components are scheduled, the results are printed in the following order:

1. List of components for manual insertion.
2. Gripper direction for each two-directional component.
3. List of auto insertion constraints.

For two-directional components, the dummy constraints and the manually-inserted component are discarded. This ends the algorithm.

The algorithm is demonstrated by solving a simple problem with a ten-component PCB. *Figure 5* shows the sample PCB and *Figure 6* shows the resulting graph. A listing of each step of the manual solution in tabular form follows. Note that for this problem, the intuitive approach of manually inserting the component with the largest degree does not lead to the optimal solution.

Sample Problem Manual Solution

Iteration 1

Step 1			Step 2			
Node	D^+	D^-	Node	D^+	D^-	E
A	2	3	A	2	3	0
B	3	2	B	2	2	0
C	4	2	C	4	2	0
D	2	2	D	2	2	1†
E	1	3	E	1	2	0
F	2	2	F	2	2	0
F′	2	2	F′	2	2	0
G	1	1	G	1	1	0
H	1	0*				
I	0	1*				
J	0	0*				

*—auto inserted
I→B, E→H auto inserted
(repeat step 1—no change)

†—manually inserted
C→D, D→C, D→E, E→D manually inserted
(G is carrier node—not used to calculate E)

Iteration 2

Step 1			Step 1		
Node	D^+	D^-	Node	D^+	D^-
A	2	3	A	2	3
B	2	2	B	2	2
C	3	1	C	3	1
E	0	1*	F	2	2
F	2	2	F′	2	2
F′	2	2	G	0	1*
G	1	1			

*—auto inserted
E→G, G→C auto inserted
(repeat step 1—no change)

Step 2			
Node	D^+	D^-	E
A	2	3	0
B	2	2	1†
C	2	1	0
F	2	2	0
F′	2	2	0

†—manually inserted
A→B, B→A, B→C, C→B manually inserted
change I→B to manually inserted

Iteration 3

Step 1			Step 2			
Node	D^+	D^-	Node	D^+	D^-	E
A	1	2	A	1	2	0
C	1	0*	F	2	1	0
F	2	2	F′	2	2	2†
F′	2	2				

*—auto inserted
F →C auto inserted
†—manually inserted
F′ →A, A →F′, dummys manually inserted

Iteration 4

Step 1		
Node	D^+	D^-
A	0	1*
F	1	0*

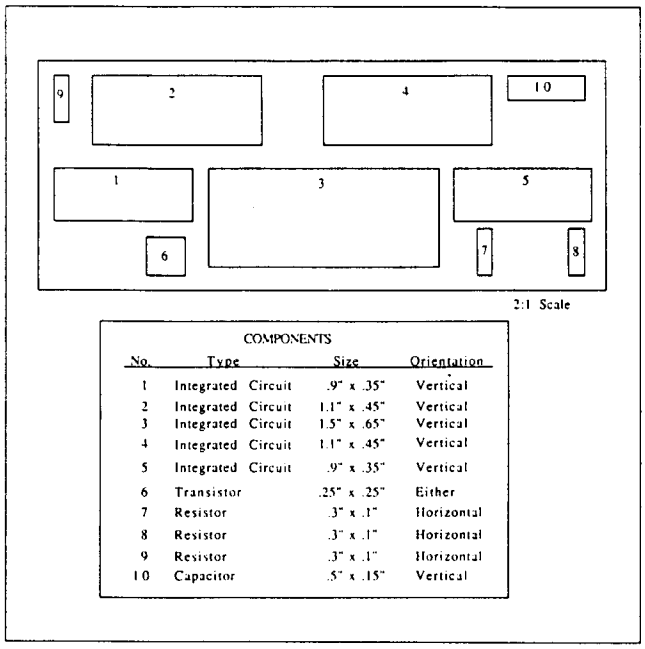
*—auto inserted
A →F auto inserted

Output:

Manually Inserted Components
B
D

Two Directional Components
F inserted horizontally

Auto Inserted Constraints
A→F
F→C
G→C
E→G
E→H



COMPONENTS

No.	Type	Size	Orientation
1	Integrated Circuit	.9" x .35"	Vertical
2	Integrated Circuit	1.1" x .45"	Vertical
3	Integrated Circuit	1.5" x .65"	Vertical
4	Integrated Circuit	1.1" x .45"	Vertical
5	Integrated Circuit	.9" x .35"	Vertical
6	Transistor	.25" x .25"	Either
7	Resistor	.3" x .1"	Horizontal
8	Resistor	.3" x .1"	Horizontal
9	Resistor	.3" x .1"	Horizontal
10	Capacitor	.5" x .15"	Vertical

Figure 5
Sample Problem PCB

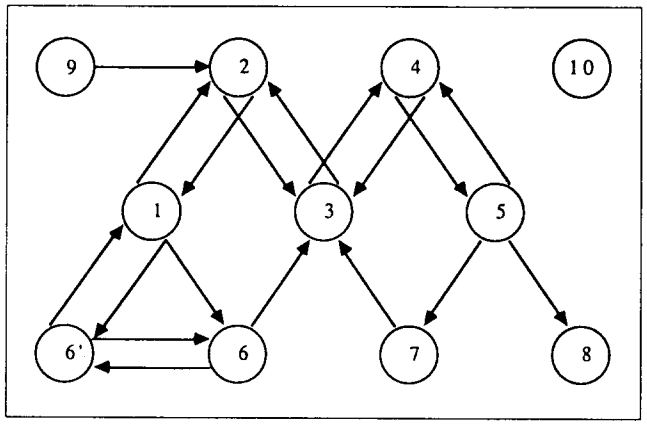


Figure 6
Sample Problem Graph
(based on gripper width of .3")

Discussion of Results

A series of experiments were run to determine the effectiveness of the algorithm. Twenty ten-component problems were randomly generated. These components were placed on a board and all feasible interferences were identified. The first ten problems activated 30% of the feasible constraints. The next ten problems activated 60% of the feasible constraints. For both sets of problems, five problems had no two-directional components and five problems had two two-directional components. The solutions obtained by the algorithm were compared to the optimal solutions. The algorithm found the optimal solution for nineteen of twenty problems. In the remaining problem, the optimum was missed by one component.

A larger problem with twenty-five nodes and forty-three constraints was analyzed in a similar manner. Once again, the algorithm was able to find the optimal solution. *Figure 7* lists the experiment results.

In theory, the worst case solution would be for a complete graph, where each component must precede every other component. For an N component problem, N-1 components would need to be manually inserted. However, in practice the graphs will generally be planar, resulting in far fewer constraints. For the problems considered here, the worst solution could have four of ten components manually inserted.

The algorithm is computationally efficient, as each step requires few calculations. Again, the worst case problem would be the complete graph and would require performing step 1 N times and step 2 N-1 times. In practice, problems with fewer constraints will require even fewer steps.

Sample Problem Solution Results

No. of Components	No. of Prec. Const.	2 Directional Components	No. of Problems	No. of Opt. Solutions	Max Difference
10	11–16	0	5	4	1
10	11–17	2	5	5	0
10	21–29	0	5	5	0
10	23–31	2	5	5	0
25	43	0	1	1	0

Figure 7

The algorithm developed here assumes that the incremental costs for manual insertion versus automatic insertion are the same for each component. If the manual and automatic insertion costs of each component are known or can be estimated, the algorithm will have to be modified to accomodate a weighted minimization problem.

The second step of the proposed process is to minimize total assembly time costs, which include travel, rotation, and insertion head change times. This is a sequencing problem that is constrained by the minimum component algorithm. One design guideline states that adding rotations can increase PCB assembly cost by nearly 25%. Also, a scattered insertion order that adds travel time can increase PCB assemby cost by 35%.[15] Adding insertion head changes may add a cost equal to manual component insertion cost. For any given PCB, it is possible that the constraints could increase assembly time costs enough to offset the reductions from the minimum component algorithm. However, using the minimum component algorithm along with the sequencing algorithm is a much simpler process expected to generate low cost PCB process plans.

Conclusion

The algorithm presented uses a graph theory approach to find the minimum number of components of a PCB board that will require manual insertion based on precedence constraints. This minimizes one major element of automated assembly cost for PCBs. The precedence constraints are usually due to interferences between a mounting head and previously inserted components, but may also be process related. The algorithm quickly gave good solutions to a series of small sample problems. It also generates a list of precedence constraints to be met by the rest of the PCB planning process. This division of the planning process into two sequential steps will reduce the complexity of the planning process for PCB assembly. It appears that this new algorithm, along with the two-step planning process, can lead to reductions in total PCB assembly cost.

References

1. R. Keeler, "Through-Hole Component Inserters: A Time of Transition," *Electronic Packaging and Production*, Vol. 27, No 6, December, 1987, pp. 42-3.
2. T.C. Chang, J. Terwilliger, "A Rule-Based System for Printed Wiring Assembly Process Planning," *International Journal for Production Research*, Vol. 25, No. 10, October, 1987, pp. 1465-82.
3. B.J. Schroer, E.F. Stafford, "Issues in Using Robotics for Electronics Assembly," *Robotics*, Vol. 2, No. 3, September, 1986, pp. 225-35.
4. T.C. Chang, R.A. Wysk, *Introduction to Automated Production Planning Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1985.
5. S.U. Randhawa, E.D. McDowell, S. Farugui, "An Integer Programming Application to Solve Sequencer Mix Problems in Printed Circuit Board Production," *International Journal of Production Research*, Vol. 23, No. 3, May/June 1985, pp. 543-52.
6. J. Cummingham, J. Browne, "A Lisp-based Heuristic Scheduler for Automatic Insertion in Electronics Assembly", *International Journal for Production Research*, Vol. 24, No. 6, Nov.-Dec., 1986, pp. 1395-1408.
7. H.P. Bao, "An Expert System for SMT Printed Circuit Board Design for Assembly," *Manufacturing Review*, Vol. 1, No. 4, December, 1988, pp. 275-80.
8. P. Cavalloro, P. Cividati, "INCA: An Expert System for Process Planning in PCB Assembly Line," *Proceeds of the Fourth Conference on Artificial Intelligence Applications*, IEEE, 1988, pp. 170-4.
9. T.C. Chang, J. Terwilliger, "PWA Planner–A Rule-Based System for Printed Wiring Assembly Process Planning," *Ninth Conference on Computers and IE*, 1987, pp. 34-8
10. G. Boothroyd, T. Shinohara, "Component Insertion Times for Electronics Assembly," *International Journal for Advanced Manufacturing Technology*, Vol. 1, No. 5, November 1986, pp. 3-18.
11. D.E. Johnson, J.R. Johnson, *Graph Theory with Engineering Applications*, Ronald Press Co., New York, 1972.
12. M.R. Garey, D.S. Johnson, *Computers and Intractability–A Guide to NP-Completeness*, W.H. Freeman and Co., New York, 1979.
13. R.M. Karp, "Reducibility Among Combinatorial Problems," *Complexity of Computer Combinations*, editors, R.E. Miller and J.W. Thatcher, Plenum Press, New York, 1972.
14. S.A. Anderson, *Graph Theory and Finite Combinatorics*, Markham Publishing Co., Chicago, 1970.
15. D. Lindsey, *The Design and Drafting of Printed Circuits*, Bishop Graphics Inc., 1979.

Author(s) Biography

Thomas Lacksonen is a PhD candidate in the Department of Industrial and Management Systems Engineering at the Pennsylvania State University. He holds an MSIE degree from South Florida University and an BSIE degree from the University of Toledo. His research interest is in the area of production planning and facilities layout. He is a member of IIE and ORSA.

Sanjay Joshi is an assistant professor in the Department of Industrial and Management Systems Engineering at the Pennsylvania State University. He holds a PhD in Industrial Engineering from Purdue University. His research interests are in Computer Aided Process Planning, Control of Manufacturing Systems and CAD applications to manufacturing. He is a senior member of IIE, SME and ASME.