A Comparison of Group and Minimum Setup Strategies in PCB Assembly

Kari Salonen, Mika Johnsson[*], Jouni Smed, Tommi Johtela, Olli Nevalainen

Turku Centre for Computer Science (TUCS) and Department of Mathematical Sciences, University of Turku, FIN-20014 Turku, Finland

Abstract

In printed circuit board (PCB) assembly, the majority of electronic components are inserted by high-speed placement machines. Although the efficient utilization of the machinery is vital for a manufacturer, it is hard to fully realize in high-mix low-volume production environments which are nowadays common in electronics assembly. On the machine level, the component setup strategy adopted by the manufacturer has a significant impact on the overall production efficiency. In this paper we compare two setup strategies proposed in the literature and review the suggested implementations. To evaluate the solutions we introduce a cost function which accounts both the number of machine setup occasions and the total number of component setup operations. Methods based on the group setup strategy turn out to yield better overall results.

Keywords: printed circuit boards, electronic components, setup strategy, group technology, sequencing

1 Introduction

The last decade has seen a rapid expansion of the production volumes in electronics manufacturing. Modern consumer goods include an ever increasing number of electronic parts, which, in turn, must be assembled more cost-effectively to ensure the competitiveness of a manufacturer. At the same time, the average product lifespan has shortened radically, and close competition forces companies to design, manufacture and market the products on a tight schedule. In this situation the actual production phase is subject to (over)ambitious goals: In addition to cost-efficiency and high-precision, flexibility is a key factor, since the same machinery is used for manufacturing slightly differing variants of the same product as well as a range of different product types. These *high-mix low-volume* environments have become common in modern electronics manufacturing and especially in *printed circuit board* (PCB) assembly.

The problems encountered in PCB assembly can be divided into four major classes according to the number of different PCBs and machines present in the problem [5]:

One PCB type and one machine (1–1) class comprises *single machine optimization* problems, which amasses feeder arrangement, placement sequencing, nozzle assignment, and component retrieval problems.

Multiple PCB types and one machine (M–1) class comprises *setup strategies for a single machine*.

One PCB type and multiple machines (1–M) class concentrates on *component allocation to multiple machines*, where the usual objective is balancing the workload of the machines in the same production line.

Multiple PCB types and multiple machines (M–M) class or *scheduling problems* concentrates on allocating jobs to lines (including routing, lot sizing and workload balancing between lines) and line sequencing (concerning duedates).

The research has traditionally concentrated on the problem class (1–1). However, the class (M–1) or *setup strategy* (i.e., the management of setups concerning multiple PCB types in a single placement machine) has also a significant impact on the efficiency. Here we can discern two kinds of setups: A *component setup* comprises the required operations to *replace one component feeder to another*. A *machine setup* comprises the required operations (component setups, conveyor belt adjustments, tooling plate changeovers, printing program updates etc.) which are required when the *manufacturing changes from one PCB type to another*.

Leon and Peters [8] classify the different setup management strategies proposed in the literature into four categories: (1) *Unique setups* consider one board at a time and specify the component–feeder assignment and the placement sequence so that the placement time for the board is minimized. (2) *Group setups* form families of similar parts so that setups are incurred only between families. (3) *Minimum setups* attempt to sequence boards and determine component–feeder assignments to minimize the changeover

[*]Corresponding author, e-mail: johnsson@cs.utu.fi, phone: +358 2 333 8671, fax: +358 2 333 8600

time. (4) *Partial setups* are characterized by the partial, rather than complete, removal of components from the machine when changing over from a product type to the next.

In this work we compare different implementations which comply either group or minimum setup strategy. Because the problem assignment described in this paper does not include machine level optimization, we do not concern partial setup strategy, which also assigns components to feeders. We discuss combining setup strategy with machine level optimization more broadly in [11]. In the *group setup strategy* the feeder assignment is determined for a group or a family of similar PCBs. Any board in this group can be produced without changing the component setup, which is only required when switching from one group to another. Because the placement time for a specific board is, in general, larger than in unique setup strategy, some efficiency can be potentially lost. *Minimum setup strategy* attempts to sequence the PCBs and determine feeder assignments to minimize the total component setup time. The idea is to perform only the feeder changes required to assemble the next PCB with no additional feeder changes or reorganization to reduce placement time. In general, similar products are produced in sequence so that little changeover time incurs.

To evaluate the solutions, we have developed a cost function based on the weighted sum of the number of component setups and the number of setup occasions. We analyze different scenarios by changing the weights and discuss the benefits and drawbacks of the strategies with respect to real-world production environments.

The remainder of this paper is organized as follows: We describe the algorithms used in our tests in Section 2. Section 3 includes the experimental results and their analysis. The concluding remarks appear in Section 4.

2 Algorithms

In this section, we recall four group setup algorithms and two minimum setup algorithms which are used in our computational experiments. We also introduce a hybrid algorithm which combines both approaches.

2.1 Group Setup Algorithms

GSA1 Group setup algorithm GSA1 [7] is a *hierarchical clustering algorithm* for grouping the boards. The algorithm uses Jaccard's similarity coefficient

$$S_{ij} = \frac{|C_i \cap C_j|}{|C_i \cup C_j|}$$

where the set C_i (C_j) contains all the components of the board i (j).

In the beginning, each board forms a singleton cluster. At the first iteration, the algorithm calculates a Jaccard's similarity coefficient for each cluster pair and merges the pair with the highest value of the coefficients into the same cluster (given that the capacity does not exceed). If the merge operation cannot be realized, the algorithm chooses the pair with the highest similarity coefficient so that the pair can be merged. After that, the similarity coefficients are updated. The iteration is continued until no improvement is possible.

The original algorithm includes also a second phase, a component-to-feeder assignment and the determination of the placement sequence. However, we omit this phase, since it deals with the placement head movements and is therefore out of the scope of this paper.

GSA2 Group setup algorithm GSA2 [9] is also based on similarity coefficients. In addition to Jaccard's similarity coefficient, the algorithm considers the component setup times and tries to group the boards so that the setup time is minimized and the machine capacity is utilized as well as possible.

The groups are formed one at a time. The algorithm computes the similarity coefficients for each board pair, and, for each board, sums up the similarity coefficients. Next, it chooses the board with the highest sum (i.e., the board that is the most "similar" to all the other boards). The board is then assigned as the first member of the first group. After that, the rest of the boards are sorted into non-increasing order according to the similarity coefficient with respect to the chosen board. Next, the algorithm tries to insert the boards in this order into the group observing the capacity and an additional threshold constraint. Finally, the boards that got chosen for the group are removed, while the rest are used in the next iteration when the algorithm forms the second group. This iteration is continued until each board has been assigned to some group.

GSA3 Group setup algorithm GSA3 [1] uses a cosine similarity coefficient and a heuristic which is based on maximum spanning trees. The algorithm allows the board to be assigned into more than one group (i.e., boards can be split). Each column of the matrix is considered as a vector in M-dimensional Euclidean space, where M is the total number of board types. Now, the cosine similarity coefficient of boards i and j is defined as the cosine of the angle between the pair of vectors that correspond to the boards

$$S_{ij} = \cos(\Theta_{ij}) = \frac{\bar{i} \cdot \bar{j}}{|\bar{i}| \cdot |\bar{j}|}.$$

The algorithm operates in two phases: First, similar boards are grouped together. After that, the algorithm decides on the basis of component setup time whether the components of a board are assigned to more than one group. The first phase begins by computing cosine similarity coefficients for each board pair. The similarity coefficients are used to construct a graph where boards are vertices and coefficients edges. By applying Prim's algorithm the graph is transformed into a maximum spanning tree. Now, the components of the new graph stand for the groups, and the algorithm examines the capacity constraint for each group. If the capacity is exceeded, the algorithm prunes the edge with the smallest weight from the initial graph, and the maximum spanning tree is reformed from this reduced graph. Edges are removed until none of the groups violates the capacity constraint. The second phase is executed if there exist groups comprising only one board, otherwise the algorithm is terminated. For each singleton group, the components of the board are assigned to a larger group in which the amount of mutual components exceed the ratio of setup times.

GSA4 Group setup algorithm GSA4 [10] uses a repair-based local search heuristic. In the first phase, the algorithm forms an initial solution with a clustering method: It begins with singular groups, and searches for group pairs that can be merged into a single group which does not exceed the capacity. It then merges the pair (i,j) which maximizes

$$\Delta C = |C_i| + |C_j| - |C_i \cup C_j|$$

where the set C_i (C_j) contains all the components of the group i (j). This is repeated until no pairs can be merged.

In the second phase, the initial solution is improved by using two local search operations: *Swap* examines all group pairs and swaps PCBs between the groups if it decreases the feeder demand. *Merge* incorporates one group to another group possibly violating the capacity constraint; the violation is then corrected by moving jobs from the group until the capacity is not exceeded.

2.2 Minimum Setup Algorithms

MSA1 Sequence dependent setup algorithm MSA1 [3] tries to sequence the boards to minimize the setup times. The algorithm uses a component communality matrix, which contains the amount of mutual components for each board pair. The goal is to maximize the amount of mutual components when a changeover from one board to another occurs. The algorithm has four variants:

- MSA1a: Component communality matrix is used iteratively to sequence the boards. The algorithm selects the board pair with the highest communality (i.e., the number of mutual components). From this pair, the board with a higher communality with some third board is placed second in the sequence (and, naturally, the remaining one will start the sequence). The third place is allocated for the board with the highest number of mutual components with the second board, and so forth until all the boards have been sequenced.

- MSA1b: The algorithm selects the board pair with the highest communality and sequences them first and second. For the third place, the algorithm selects the board with the highest communality with the previous boards. This is repeated until all the boards are sequenced.

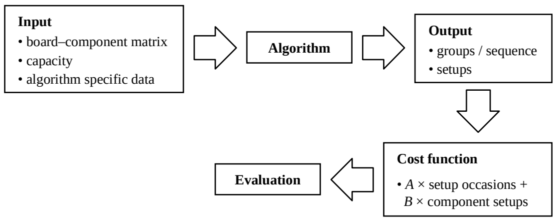

Figure 1: The test arrangement comprises separate computation and evaluation phases

- MSA1c: This variant resembles MSA1a but uses a percentage component communality matrix, which is constructed from the component communality matrix by dividing each value by the total number of components assembled to the corresponding board.

- MSA1d: This variant resembles MSA1b but uses a percentage component communality matrix.

MSA2 Sequence dependent setup algorithm MSA2 [4] uses an upper-bound heuristic to sequence boards and then applies *keep tool needed soonest* (KTNS) method [12] to select which components are removed when a changeover occurs. Sequencing is done in two phases: First, the boards are sequenced so that each successive board has the minimum number of new components in comparison to its predecessor. After that, the sequence is improved by applying a 2-opt heuristic.

2.3 A Hybrid Algorithm

SGSA Sequenced group setup algorithm SGSA is a hybridization of group and minimum setup algorithms. It uses GSA1 to group the boards and then sequences the groups with the algorithm MSA1a.

3 Computational Experiments

The test runs comprise two phases (see Figure 1): First, we let each algorithm to solve the processing sequence (which also indicates the grouping) of PCBs for a given problem. After that, we use a cost function to evaluate the results.

Our cost function is

$$\text{cost}_{A,B} = A \cdot \text{setup\_occasions} + B \cdot \text{component\_setups} \quad (1)$$

where the parameters A and B can be viewed as the time factors for starting to set up components and setting up an individual component, respectively. In PCB assembly, a single component feeder can be changed in 1–5 minutes but it may take, for example, 15 minutes to prepare the machine for the component setup operations and to take it back on line when the setup is complete. By setting $B = 0$ we can compare the algorithms by the number of machine setup occasions (i.e., a job grouping problem), and by setting $A = 0$

we can make a comparison on the basis of the total number of component changeovers (i.e., a tool switching problem; cf. [2]). In our experiments, we compare the values of (1) for $B = 1$ and $A = 0$, 10, 20, and 30.

We must emphasize that some of the tested algorithms are not originally designed to meet our evaluation criterion. Usually the algorithms are evaluated solely on the basis of component changeovers. However, our objectives emerge from a practical point of view. If these algorithms are to be used in real-world production environments, they should observe both the number of setups and the number of setup occasions.

Our test data originates from two different sources. The first data set contains three documented test problems from prior research[1]: matr853 from Shtub and Maimon [9], matr930 from Bhaskar and Narendran [1], and 21pcb from Smed *et al.* [10]. The second data set is drawn from a real-world production data provided by our partnership company. This set contains 10 sample problems, which are generated by selecting 20 or 50 jobs randomly from the company's production program.

To evaluate the effect of the feeder capacity, we vary it as follows: The feeder capacity is 20 for matr930, 25 for matr853, and 160 for 21pcb. For the second problem set, we use capacities 80 and 160. In GSA3, the setup time for PCB is set to 6 and the setup time for a component to 1.

Figures 2, 3 and 4 contain the evaluation of solutions for matr853, matr930 and 21pcb. Each of the three cases show similar tendencies. For matr853, SGSA and GSA1–4 all find equally good solutions. The MSA1 variants are clearly weaker for all the settings of A, and MSA2 provides always slightly better results. For matr930, the group algorithms are again most profitable. Interestingly, MSA2 finds the best solution for $A = 0$, but loses to group setup algorithms if machine setups are also concerned. Similar observations can be made also in the case 21pcb, apart from GSA3 which performs now somewhat weaker than SGSA, GSA1 and GSA2. Also, the overall performance of GSA4 appears to be the best because of the larger problem size.

Figure 5 summarizes the average results for the case of 20 PCBs drawn from industrial data for capacities 80 and 160. When the capacity is set to 80, MSA2 shows its power in sequencing (i.e., when $A = 0$). With a higher capacity, algorithms SGSA, GSA1, GSA2 and GSA4 narrow the gap and find almost equally good solutions for $A = 0$ as MSA2. MSA1c seems to dominate the other MSA1 variants but, generally speaking, MSA1 gives the worst results in every parameter configuration. Among the pure group setup methods, GSA3 gives the best results when the capacity is 80, but when the capacity is increased, the results of GSA3 do not improve significantly—in fact, it gives the worst results. Figure 6 compares the algorithms in the case of 50 PCBs for capacities 80 and 160, and the similarity of the results seems to further confirm the observations made in the previous case.

[1]The test data is available in the web: `http://www.cs.utu.fi/scheduling/testdata/`

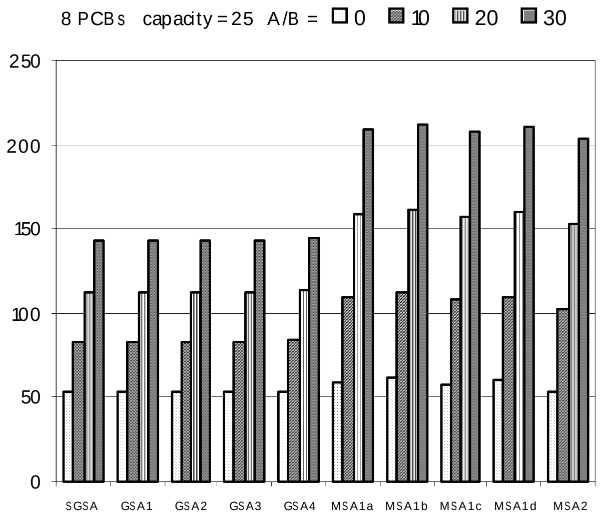

Figure 2: Evaluation of the results for the case matr853

To summarize, we can make the following observations from the computational results:

1. In general, the solutions of MSA2 yield the lowest values of the cost function when $A = 0$ (i.e., when we are solving a pure tool switching problem). However, the difference to SGSA, GSA1, GSA2 and GSA4 is surprisingly small.

2. If one concerns also machine setups (i.e., wants to avoid unnecessary setup occasions), MSA2 is no longer the obvious choice but the group setup methods become more preferable.

3. Apart from GSA3 and GSA4, an increase in the capacity does not change the mutual preference of the methods.

4. The results for the four MSA1 variants are weaker than for the other methods of this study. A possible reason for this is that MSA1 does not benefit from the extra components left over from the previous setup(s). The algorithm considers all the components of the previous setup that are not mutual with the current setup to be removed, even though it is possible that some of them could be preserved for the next setup.

5. The results of the group setup algorithms do not differ significantly from each other.

6. GSA3 provides good results for smaller feeder capacities, because—unlike the other algorithms—it allows the components of a PCB to be divided into two or more groups if that turns out be beneficial. However,

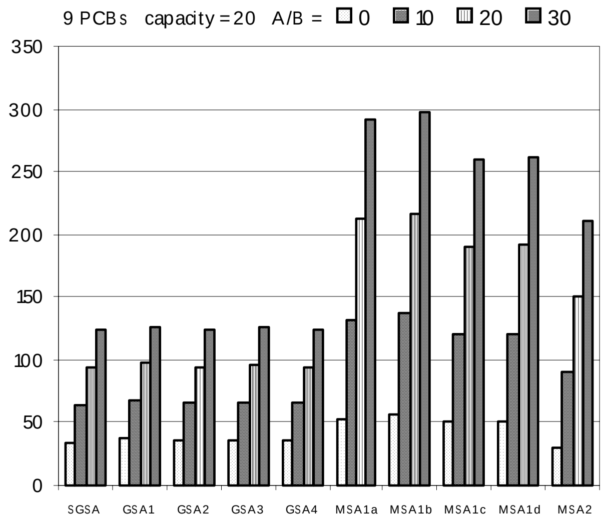

Figure 3: Evaluation of the results for the case matr930

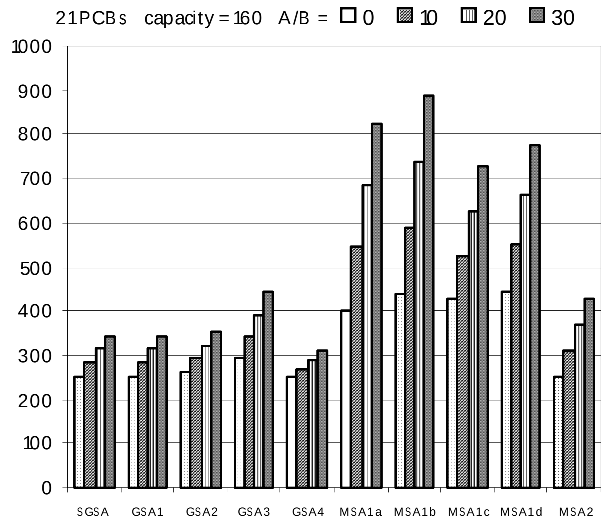

Figure 4: Evaluation of the results for the case pcb21

this is rarely desirable in real-world production environments. When the capacity is increased, GSA3 performs worse than the group setup algorithms on average.

7. Among the group setup algorithms, GSA4 provides the best average results.

8. As expected, SGSA yields always better results than GSA1. However, this margin decreases when the capacity is increased, since the number of groups decreases.

9. If the solution consists of more than three groups, the solution can be further improved by sequencing the groups to minimize the number component setups (like in SGSA). However, there may be practical considerations against this hybridization.

10. Each algorithm provides solutions in a reasonable time. For instance, in the case of 50 PCBs, the running times on a 200 MHz PC vary from a few seconds to less than one minute.

4 Final Remarks

In this paper we compared methods presented in the literature for determining setups for a set of boards on a single machine. In particular, we studied two strategies—group and sequence dependent—which are commonly suggested. In order to compare these strategies, we introduced a cost function which corresponds to the practical considerations of real-world production. Test cases were derived from literature and actual production data. We observed that group strategy, in general, gives better solutions.

Apart from our experiments, there are also practical reasons why group setup strategy suits well in high-mix low-volume production environments. In group setup strategy, smaller production batch sizes become economical, which enables to cut down the work-in-process (WIP) levels. Although the unique setup strategy enables one to construct better placement sequences for each PCB—and hence the printing time of each individual PCB can be shorter than in the group setup—the overall production time can be considerably longer, because setups occur whenever the produced PCB type changes. The possible theoretical advances of minimum and partial setup strategy are outweighed by the practical benefits of the group setup strategy: Because setups, albeit larger than in other setup strategies, occur less frequently, the human operator who carries out the component changeovers is less prone to make mistakes, and thus the economical risks involved in the setup operations diminish. The human operator usually prefers to change ten components once than to change one component ten times. Moreover, group setup strategy allows to design a production planning system, which provides the production planner with more freedom, since the production sequence among the groups as well as within an individual group can be easily altered [11]. Group setup strategy can be also extended to account multiple criteria by using fuzzy sets [6].

Acknowledgments

The authors would like to thank Teleste Corporation (Nousiainen, Finland) for kindly providing production data for our test use. The support from Mr. Jouni Lehtinen is especially acknowledged.

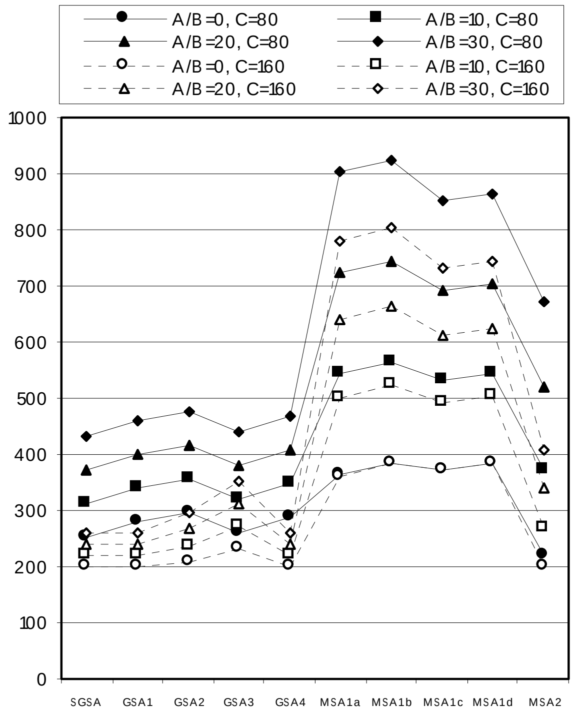

Figure 5: Evaluation of the results for the case of 20 PCBs selected from industrial data. Solid points and lines indicate capacity 80; open points and dashed lines capacity 160.

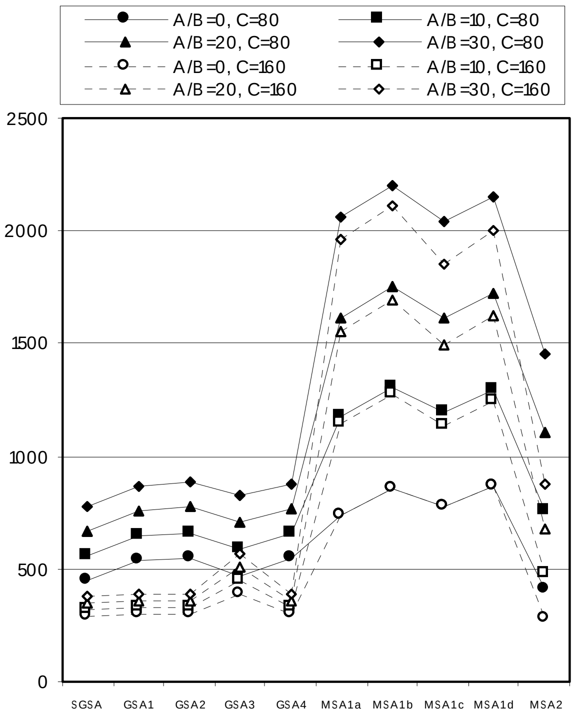

Figure 6: Evaluation of the results for the case of 50 PCBs selected from industrial data (for explanation, see Figure 5).

References

[1] G. Bhaskar and T. T. Narendran. Grouping PCBs for set-up reduction: A maximum spanning tree approach. *International Journal of Production Research*, 34(3):621–32, 1996.

[2] Y. Crama and J. van de Klundert. The approximability of tool management problems. Technical Report RM 96034, Maastricht Economic Research School on Technology and Organisation, 1996.

[3] S. Dillon, R. Jones, C. J. Hinde, and I. Hunt. PCB assembly line setup optimization using component commonality matrices. *Journal of Electronics Manufacturing*, 8(2):77–87, 1998.

[4] H. O. Günther, M. Gronalt, and R. Zeller. Job sequencing and component set-up on a surface mount placement machine. *Production Planning & Control*, 9(2):201–11, 1998.

[5] M. Johnsson. *Operational and Tactical Level Optimization in Printed Circuit Board Assembly*. PhD thesis, University of Turku, 1999. TUCS Dissertations 16.

[6] T. Johtela, J. Smed, M. Johnsson, and O. Nevalainen. Fuzzy approach for modeling multiple criteria in the job grouping problem. Technical Report 227, Turku Centre for Computer Science, Dec. 1998.

[7] V. J. Leon and B. A. Peters. Replanning and analysis of partial setup strategies in printed circuit board assembly systems. *International Journal of Flexible Manufacturing Systems*, 8(4):389–412, 1996.

[8] V. J. Leon and B. A. Peters. A comparison of setup strategies for printed circuit board assembly. *Computers & Industrial Engineering*, 34(1):219–34, 1998.

[9] A. Shtub and O. Maimon. Role of similarity in PCB grouping procedures. *International Journal of Production Research*, 30(5):973–83, 1992.

[10] J. Smed, M. Johnsson, M. Puranen, T. Leipälä, and O. Nevalainen. Job grouping in surface mounted component printing. *Robotics and Computer-Integrated Manufacturing*, 15(1):39–49, 1999.

[11] J. Smed, T. Johtela, M. Johnsson, M. Puranen, and O. Nevalainen. An interactive system for scheduling jobs in electronic assembly. Forthcoming in *International Journal of Advanced Manufacturing Technology*, 1999.

[12] C. S. Tang and E. V. Denardo. Models arising from a flexible manufacturing machine. *Operations Research*, 36(5):767–84, 1988.