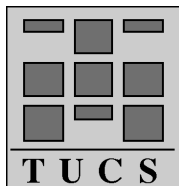


A Comparison of Group and Minimum Setup Strategies in PCB Assembly

**Jouni Smed
Kari Salonen
Mika Johnsson
Tommi Johtela
Olli Nevalainen**



**Turku Centre for Computer Science
TUCS Technical Report No 327
January 2000
ISBN 952-12-0608-X
ISSN 1239-1891**

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. We conclude that methods based on the group setup strategy yield better overall results.

Keywords: printed circuit boards, electronics assembly, setup strategy, product sequencing, group technology

TUCS Research Group
Algorithmics

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 to manufacture 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.

Generally speaking, 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 (Johnsson, 1999):

- *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 due dates).

The class (M–1) or *setup strategy* (i.e., the management of setups concerning multiple PCB types in a single placement machine) has a significant impact on the overall 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 (1998) classify the different setup management strategies into four categories:

- *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. This is a common strategy when dealing with a single product and a single machine in a high-volume production environment (Ball and Magazine, 1988; Leipälä and Nevalainen, 1989; Crama *et al.*, 1990; Laarhoven and Zijm, 1993; Zijm and Harten, 1993; Bard *et al.*, 1994; Crama *et al.*, 1997).
- *Group setups* form families of similar parts so that setups are incurred only between families.
- *Minimum setup* strategy attempts to sequence boards and determine component–feeder assignments to minimize the changeover time.
- *Partial setup* strategy also attempts to sequence the boards as the minimum setup but it also considers reorganizing the feeders to reduce placement time (Leon and Peters, 1996, 1998; Peters and Subramanian, 1996). Because the goal is often to minimize the makespan, the partial setup strategy resides between the unique setup strategy (where only the placement time for each individual PCB is minimized) and the minimum setup strategy (where only the changeover time of each PCB is minimized).

In this work we compare different implementations which comply either with 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 (in fact, minimum setup would dominate partial setup). To evaluate the solutions, we introduce 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 begin with a review of related work on group and minimum setup strategies. After that, we describe the algorithms used in our tests and present the experimental results and their analysis in separate sections. Next, we propose an improved grouping algorithm based on the observations made from the results. Finally, we close the paper with concluding remarks.

Related Work on Setup Strategies

Group Setup Strategy

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 the unique setup strategy, some efficiency can be potentially lost. There are variations of the group setup strategy, where a certain set of common or standard components are left on the machine, while the rest of the components (residual or custom) are added or removed as required for a particular board (Smed *et al.*, 1999).

Carmon *et al.* (1989) describe a group setup (GSU) method for a high-mix low-volume production environment. In GSU, the products are divided into groups, each of which is produced in two stages: First, set up the *common components* and insert them to the whole group, and, after that, set up the *residual components* and insert them on each PCB separately. Maimon *et al.* (1993) compare GSU to sequence dependent scheduling (SDS) on three performance measures—line throughput, average WIP level and implementation complexity—and conclude that in general SDS performs better on the last two areas.

Maimon and Shtub (1991) present a *mixed-integer programming* formulation and a heuristic method for grouping a set of PCBs to minimize the total setup time. A user-defined parameter is used to indicate, whether multiple loading of PCBs and components is allowed (cf. partition and repeat strategy of McGinnis *et al.*, 1992). This approach is developed further by Daskin *et al.* (1997). Here the goal is to minimize the total component and PCB loading costs subject to a capacity constraint. The authors present a mathematical formulation for the PCB-grouping problem, show that it is \mathcal{NP} -complete, and give a *branch-and-bound* based heuristic algorithm for solving it.

Shtub and Maimon (1992) establish that grouping PCBs is an extension of the *set-covering problem* and present a general heuristic approach based on *cluster analysis* and *similarity measures*, which are traditionally found in the group technology literature. Here the goal is to minimize the total production time, but since insertion times are assumed to be constant, the objective reduces to minimizing the total setup time of the groups.

Hashiba and Chang (1992) study one machine case when the objective is to minimize the number of setups. They decompose the setup problem into three subproblems—grouping PCB types, sequencing the groups, and assigning components for jobs—and apply heuristic algorithms to each of

them individually. Furthermore, the authors experiment with a *simulated annealing* method and observe that it gives better solutions than the heuristic decomposition approach.

Bhaskar and Narendran (1996) apply *graph theory* for solving the grouping problem of PCBs. The boards are modeled as nodes of a graph and their similarities as the arcs between the nodes. After that, a *maximum spanning tree* is constructed for identifying the PCB groups.

Smed *et al.* (1999) discuss the *job grouping problem* in PCB assembly. The authors give an integer programming formulation of the problem, and compare several heuristic algorithms based on greedy, clustering and repair-based local search methods. Johtela *et al.* (1998) expand the problem to account multiple and possibly conflicting job grouping criteria—such as different board widths, adhesive types, and production priorities—by modeling them as *fuzzy sets*.

Minimum Setup Strategy

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.

Barnea and Sipper (1993) consider a case of one machine and recognize two subproblems: *sequence and mix*. They present a mathematical model of the problem and use a heuristic approach based on KTNS (*keep tool needed soonest*) policy introduced by Tang and Denardo (1988). In each iteration, the algorithm generates a new partial sequence by using a sequencing algorithm (which decides the next job to enter the sequence) and a mix algorithm (which updates the component mix with KTNS).

Jain *et al.* (1996) present a four-stage method for optimizing the setup: Firstly, a greedy heuristic maximizing the component similarity of the jobs is used to determine an initial processing sequence. Secondly, the components are assigned to feeders according to KTNS policy. Thirdly, the jobs are rearranged by applying *2-opt heuristic* and KTNS. Finally, because the production is a continuous process, at the end of the sequence the frequently used components are preserved for the next production period (i.e., the approach heeds the rolling horizon framework).

Günther *et al.* (1998) present a typical surface mount technology production line and apply the minimum setup strategy approach, in which the PCBs are sequenced so that each subsequent PCB has the maximum number

of components common with its predecessor. The authors discern three different subproblems—job sequencing, component setup and feeder assignment—and solve each of them with heuristic algorithms.

Dillon *et al.* (1998) discuss minimizing the setup time by sequencing PCBs on a surface mount technology production line. The authors present four variants of a *greedy heuristic* which aims at maximizing iteratively the component commonality whenever the PCB type changes. This is realized by using a component commonality matrix from which board pairs with a high number of common components can be identified.

T. T. Narendran and coworkers have studied a heuristic approach which starts from an initial setup (a seed) and sequences the PCBs by looking for the similarities between the current setup and the PCBs remaining to be sequenced. Rajkumar and Narendran (1998) form the sequence by considering the overall component requirements of a particular PCB and the number of extra components required in the current setup. Kumar and Narendran (1997) add the slack time of the PCB as a third constraint in the problem formulation.

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 new hybrid algorithm which combines both approaches.

Group Setup Algorithms

GSA1 Group setup algorithm GSA1 (Leon and Peters, 1996) 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 the components of the board i (j).

Each board forms initially a singleton cluster. At the first iteration, the algorithm calculates Jaccard's similarity coefficient for each cluster pair and merges the pair with the highest coefficient value (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 merge is feasible. After that, the similarity coefficients are updated. The process is iterated 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 the present paper.

GSA2 *Iterative group setup* algorithm GSA2 (Shtub and Maimon, 1992) 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 efficiently.

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 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 coefficients 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 (Bhaskar and Narendran, 1996) 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 board–component 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

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

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 coefficients are used to construct a graph where boards are vertices and coefficients edges. By applying Prim’s algorithm (Cormen

et al., 1990) the graph is transformed into a maximum spanning tree. Now, the connected 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 exceeds the ratio of setup times.

GSA4 Group setup algorithm GSA4 (Smed *et al.*, 1999) 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 without exceeding the feeder capacity. It then merges the feasible 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 component types of the group i (j). This operation 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 possibly violating the capacity constraint; the violation is then corrected by moving jobs from the unfeasible group to the other groups until the capacity constraint is met.

Minimum Setup Algorithms

MSA1 Sequence dependent setup algorithm MSA1 (Dillon *et al.*, 1998) sequences 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.
- 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 (Günther *et al.*, 1998) uses an upper-bound heuristic to sequence boards and then applies *keep tool needed soonest* (KTNS) method of Tang and Denardo (1988) 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*.

A Hybrid Algorithm

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

Computational Experiments

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

$$\text{cost}_{A,B} = A \cdot \text{setup_occasions} + B \cdot \text{component_setups} \quad (1)$$

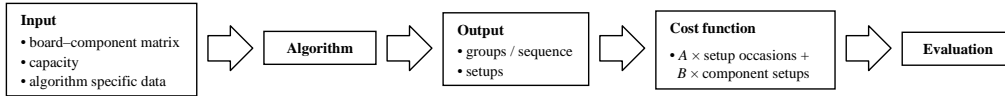


Figure 1: The test arrangement comprises separate computation and evaluation phases

where the parameter A is the time factor for starting to set up the components and B is the time factor for setting up an individual component. 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. Crama and Klundert, 1996). In our experiments, we compare the values of equation 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: **matr853** from Shtub and Maimon (1992), **matr930** from Bhaskar and Narendran (1996), and **21pcb** from Smed *et al.* (1999) (the test data is available in the web: <http://www.cs.utu.fi/scheduling/testdata/>). 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. We assume that each component takes only one feeder slot, although in reality the width of the component may vary (e.g., method GSA4 allows different component widths). In GSA3, the setup time for PCB is set to 6 and the setup time for a component to 1.

Table 1 contains the evaluation of solutions for **matr853**, **matr930** and **21pcb**. The results for these problems show similar tendencies. For **matr853**, SGSA (hybrid algorithm) and GSA1–3 (hierarchical clustering, iterative setup,

Method	matr853				matr930				pcb21			
	<i>A / B</i>				<i>A / B</i>				<i>A / B</i>			
	0	10	20	30	0	10	20	30	0	10	20	30
SGSA	53	83	113	143	34	64	94	124	254	284	314	344
GSA1	53	83	113	143	37	67	97	127	254	284	314	344
GSA2	53	83	113	143	35	65	95	125	262	292	322	352
GSA3	53	83	113	143	36	66	96	126	292	342	392	442
GSA4	54	84	114	144	35	65	95	125	250	270	290	310
MSA1a	59	109	159	209	52	132	212	292	403	543	683	823
MSA1b	62	112	162	212	57	137	217	297	438	588	738	888
MSA1c	58	108	158	208	50	120	190	260	426	526	626	726
MSA1d	60	110	160	210	51	121	191	261	443	553	663	773
MSA2	53	103	153	203	30	90	150	210	250	310	370	430

Table 1: Evaluation of the results for the case **matr853**, **matr930** and **pcb21**. The table contains the values of cost function (1) for different (A, B) combinations.

and maximum spanning tree) all find equally good solutions. The MSA1 (communality matrix) variants are clearly weaker for all the settings of A , and MSA2 (sequence dependent setup with KTNS) 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. However, both of these test cases are quite small and not very realistic in modern production environments. Similar observations can be made in the larger case **21pcb**, apart from GSA3 which performs now somewhat weaker than SGSA, GSA1 and GSA2. Also, the overall performance of GSA4 (repair-based local search) appears to be the best because of the larger problem size.

Figure 2 summarizes the average results for the case of 20 PCBs drawn from industrial data for capacities 80 and 160. When the capacity is 80, MSA2 shows again its power in sequencing (i.e., when $A = 0$). With a higher capacity, algorithms SGSA, GSA1, GSA2 and GSA4 become more effective and find almost equally good solutions for $A = 0$ as MSA2. MSA1c (percentage communality matrix) seems to dominate the other MSA1 variants but, generally speaking, MSA1 gives the worst results with all parameter configurations. 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 3 compares the algorithms in the case of 50 PCBs for capacities 80

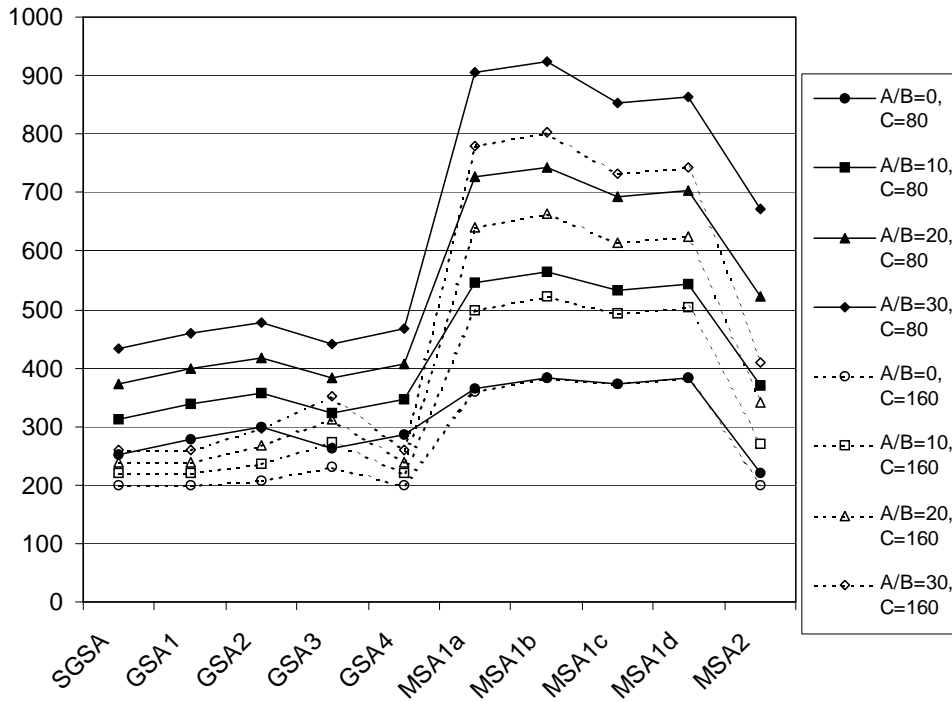


Figure 2: Evaluation of the results for the case of 20 PCBs selected from industrial data. Solid points and lines indicate capacity 80; open points and dashed lines capacity 160. The values of cost function (1) are on the y-axis.

and 160, and the similarity of the results seems to further confirm the observations made in the previous case.

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

1. In general, the solutions of MSA2 (sequence dependent setup with KTNS) 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.

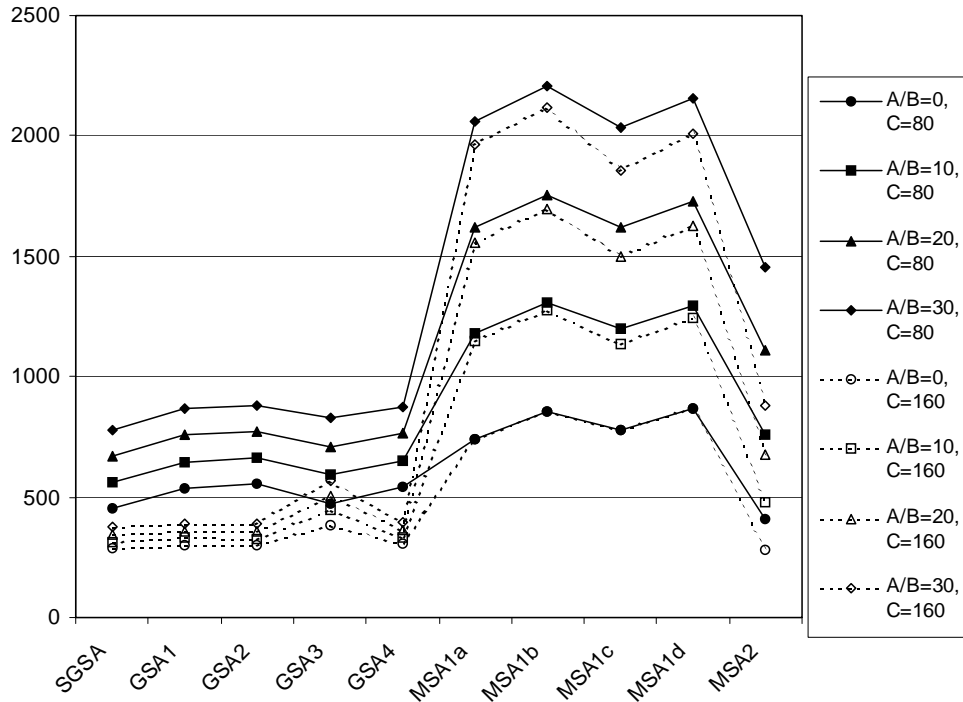


Figure 3: Evaluation of the results for the case of 50 PCBs selected from industrial data (for explanation, see Figure 2).

4. The results for the four MSA1 variants (which use component commu-
nality matrices) 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 (maximum spanning tree) provides good results for smaller feeder
capacities, because—unlike the other algorithms—it allows the compo-
nents of a PCB to be divided into two or more groups if that turns out
be beneficial. However, 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 (repair-based local search)

provides the best average results.

8. As expected, SGSA yields always better results than GSA1 (hierarchical clustering). 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.

Improved Grouping Algorithm

In the method GSA4, the initial grouping is done by merging groups for which the intersection of the component types is maximal. Conversely, method GSA1 uses Jaccard's similarity coefficient for selecting the groups to be merged. Hence, GSA1 favors group pairs with a large number of common component types and a small total number of component types, whereas GSA4 does not discriminate the groups with a large selection of different component types. When we analyzed the computational results, we observed that the groupings found by these methods are quite dissimilar, because the methods rarely end up choosing the same group pairs. Due to this observation we changed the method GSA4 so that it uses the result of GSA1 as the initial solution and after that performs the local search operations to improve it. We call this modified method GSA4b. To solve whether GSA4b gives better results than GSA4, we conducted an additional set of tests. The test problems were formed by selecting several cases of test sets of size 50, 100 or 150 PCBs from real-world production data and by setting the capacity constraint to 80 or 112 feeders (112 is the greatest number of components in a single product among the chosen boards). Table 2 gives the results for one test set configuration, where there are 100 boards and the feeder capacity is 112. This test set is a typical example of the results in general, since there are cases where GSA4 gives better results than GSA4b and vice versa.

This observation forms the ground for a *multistart grouping algorithm* (MSGGA) which combines methods GSA4 and GSA4b. The final solution is the one with the smaller number of groups. In Figure 4 the results for different test sets are summarized by comparing the average results of each

case	Method				Overall
	GSA1	GSA4	GSA4b	MSGA	minimum
1	11	10	10	10	10
2	9	9	9	9	9
3	11	10	10	10	10
4	10	11	10	10	10
5	11	11	11	11	11
6	9	9	8	8	8
7	11	11	10	10	10
8	10	10	10	10	10
9	13	12	12	12	12
10	11	11	11	11	11
11	10	10	10	10	10
12	11	11	11	11	11
13	10	10	10	10	10
14	10	9	10	9	9
15	11	10	10	10	10
16	10	10	10	10	10
17	11	11	11	11	11
18	10	9	10	9	9
19	12	12	12	12	12
20	10	10	9	9	9
sum	211	206	204	202	202
average	10,55	10,3	10,2	10,1	10,1

Table 2: Results of the test cases with 100 boards and 112 capacity

test set to the overall minimum result of the same set. If we now compare the original GSA1 to the algorithm MSGA, we note that MSGA gives significantly better results. MSGA also dominates GSA4 and GSA4b, which is expected since it incorporates them both. To summarize we propose the following method, MSGA, to be an efficient grouping algorithm:

First phase Generate an initial grouping by merging iteratively pairs which maximize $|C_i| + |C_j| - |C_i \cup C_j|$. Improve the initial grouping by using local search operations swap and merge. Let the solution be S_1 . (GSA4)

Second phase Generate an initial grouping by merging iteratively pairs which maximize $|C_i \cap C_j|/|C_i \cup C_j|$. Improve the initial grouping by

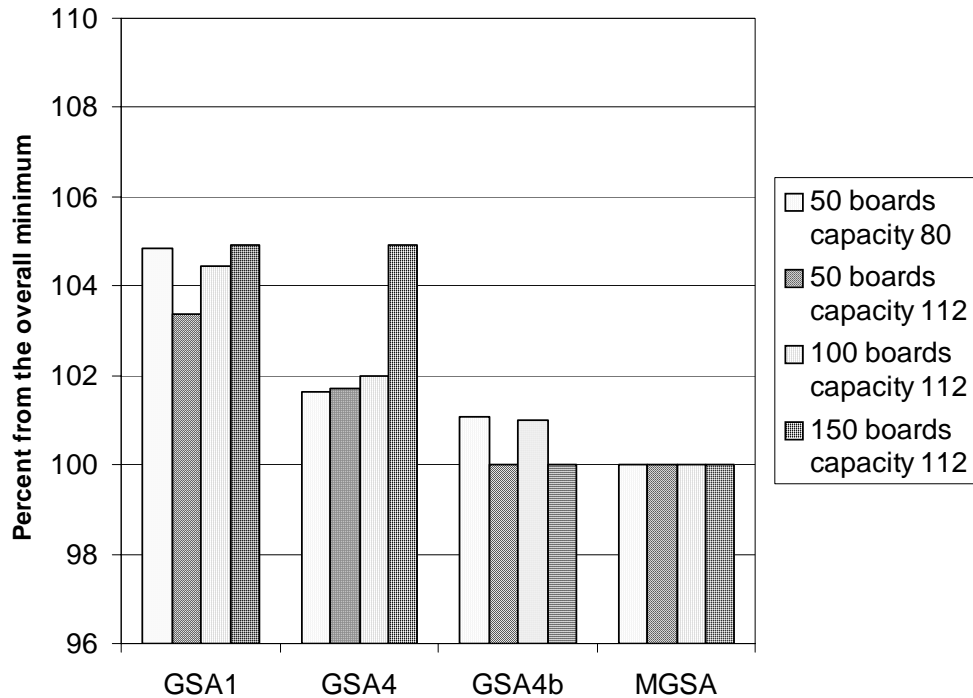


Figure 4: A comparison of the methods GSA1, GSA4, GSA4b and MGSA. The results indicate for the number of the groups the percent difference from the overall minimum.

using local search operations swap and merge. Let the solution be S_2 . (GSA4b)

Evaluation Compare S_1 and S_2 and let the better one be the final solution.

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 minimum—which are commonly suggested. 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 setup strategy, in general, gives better solutions. Among the group setup methods, the improvement methods seemed to give better results than the

constructive methods. Based on our computational results, we proposed a new multistart grouping algorithm which utilizes the versatility observed in the different grouping criteria.

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. A 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 (Smed *et al.*, 2000). Group setup strategy can be also extended to account multiple criteria by using fuzzy sets (Johtela *et al.*, 1998).

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.

References

- Ball, M. O., and Magazine, M. J. (1988). “Sequencing of insertions in printed circuit board assembly.” *Operations Research*, 36(2), 192–201.
- Bard, J. F., Clayton, R. W., and Feo, T. A. (1994). “Machine setup and component placement in printed circuit board assembly.” *International Journal of Flexible Manufacturing Systems*, 6, 5–31.
- Barnea, A., and Sipper, D. (1993). “Set-up reduction in PCB automated assembly.” *Computer Integrated Manufacturing Systems*, 6(1), 18–26.

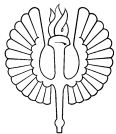
- Bhaskar, G., and Narendran, T. T. (1996). “Grouping PCBs for set-up reduction: A maximum spanning tree approach.” *International Journal of Production Research*, 34(3), 621–32.
- Carmon, T. F., Maimon, O. Z., and Dar-El, E. M. (1989). “Group set-up for printed circuit board assembly.” *International Journal of Production Research*, 27(10), 1795–810.
- Cormen, T. H., Leiserson, C. E., and Rivest, R. L. (1990). *Introduction to algorithms*. Cambridge, MA: MIT Press.
- Crama, Y., Flippo, O. E., Klundert, J. van de, and Spieksma, F. C. R. (1997). “The assembly of printed circuit boards: A case with multiple machines and multiple board types.” *European Journal of Operational Research*, 98(3), 457–72.
- Crama, Y., and Klundert, J. van de. (1996). *The approximability of tool management problems* (Tech. Rep. No. RM 96034). Maastricht Economic Research School on Technology and Organisation.
- Crama, Y., Koleen, A. W. J., Oerlemans, A. G., and Spieksma, F. C. R. (1990). “Throughput rate optimization in the automated assembly of printed circuit boards.” *Annals of Operations Research*, 26, 455–80.
- Daskin, M. S., Maimon, O., Shtub, A., and Braha, D. (1997). “Grouping components in printed circuit board assembly with limited components staging capacity and single card setup: Problem characteristics and solution procedures.” *International Journal of Production Research*, 35(6), 1617–38.
- Dillon, S., Jones, R., Hinde, C. J., and Hunt, I. (1998). “PCB assembly line setup optimization using component commonality matrices.” *Journal of Electronics Manufacturing*, 8(2), 77–87.
- Günther, H. O., Gronalt, M., and Zeller, R. (1998). “Job sequencing and component set-up on a surface mount placement machine.” *Production Planning & Control*, 9(2), 201–11.
- Hashiba, S., and Chang, T. C. (1992). “Heuristic and simulated annealing approaches to PCB assembly setup reduction.” In G. J. Olling and F. Kimura (Eds.), *Human aspects in computer integrated manufacturing. IFIP transactions b-3* (pp. 769–77). Amsterdam, The Netherlands: North Holland, Elsevier Science Publishers.

- Jain, S., Johnson, M. E., and Safai, F. (1996). “Implementing setup optimization on the shop floor.” *Operations Research*, 43(6), 843–51.
- Johnsson, M. (1999). *Operational and tactical level optimization in printed circuit board assembly*. PhD thesis, University of Turku. (TUCS Dissertations 16)
- Johtela, T., Smed, J., Johnsson, M., and Nevalainen, O. (1998). *Fuzzy approach for modeling multiple criteria in the job grouping problem* (Tech. Rep. No. 227). Turku Centre for Computer Science.
- Kumar, K. R., and Narendran, T. T. (1997). “A heuristic for sequencing PCBs with due-dates.” *International Journal of Operations & Production Management*, 17(5), 446–67.
- Laarhoven, P. J. M. van, and Zijm, W. H. M. (1993). “Production preparation and numerical control in PCB assembly.” *International Journal of Flexible Manufacturing Systems*, 5, 187–207.
- Leipälä, T., and Nevalainen, O. (1989). “Optimization of the movements of a component placement machine.” *European Journal of Operational Research*, 38, 167–77.
- Leon, V. J., and Peters, B. A. (1996). “Replanning and analysis of partial setup strategies in printed circuit board assembly systems.” *International Journal of Flexible Manufacturing Systems*, 8(4), 389–412.
- Leon, V. J., and Peters, B. A. (1998). “A comparison of setup strategies for printed circuit board assembly.” *Computers & Industrial Engineering*, 34(1), 219–34.
- Maimon, O., and Shtub, A. (1991). “Grouping methods for printed circuit boards.” *International Journal of Production Research*, 29(7), 1370–90.
- Maimon, O. Z., Dar-El, E. M., and Carmon, T. F. (1993). “Set-up saving schemes for printed circuit boards assembly.” *European Journal of Operational Research*, 70(2), 177–90.
- McGinnis, L. F., Ammons, J. C., Carlyle, M., Cranmer, L., DePuy, G. W., Ellis, K. P., Tovey, C. A., and Xu, H. (1992). “Automated process planning for printed circuit card assembly.” *IIE Transactions*, 24(4), 18–30.

- Peters, B. A., and Subramanian, G. S. (1996). “Analysis of partial setup strategies for solving the operational planning problem in parallel machine electronic assembly systems.” *International Journal of Production Research*, 34(4), 999–1021.
- Rajkumar, K., and Narendran, T. T. (1998). “A heuristic for sequencing PCB assembly to minimize set-up times.” *Production Planning & Control*, 9(5), 465–76.
- Shtub, A., and Maimon, O. (1992). “Role of similarity in PCB grouping procedures.” *International Journal of Production Research*, 30(5), 973–83.
- Smed, J., Johnsson, M., Puranen, M., Leipälä, T., and Nevalainen, O. (1999). “Job grouping in surface mounted component printing.” *Robotics and Computer-Integrated Manufacturing*, 15(1), 39–49.
- Smed, J., Johtela, T., Johnsson, M., Puranen, M., and Nevalainen, O. (2000). *An interactive system for scheduling jobs in electronic assembly*. (Forthcoming in *International Journal of Advanced Manufacturing Technology*)
- Tang, C. S., and Denardo, E. V. (1988). “Models arising from a flexible manufacturing machine.” *Operations Research*, 36(5), 767–84.
- Zijm, W. H. M., and Harten, A. van. (1993). “Process planning for a modular component placement system.” *International Journal of Production Economics*, 30–31, 123–35.

Turku Centre for Computer Science
Lemminkäisenkatu 14
FIN-20520 Turku
Finland

<http://www.tucs.fi>



University of Turku
• **Department of Mathematical Sciences**



Åbo Akademi University
• **Department of Computer Science**
• **Institute for Advanced Management Systems Research**



Turku School of Economics and Business Administration
• **Institute of Information Systems Science**