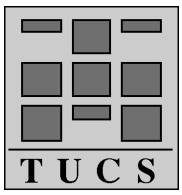
Job Grouping in Surface Mounted Component Printing

Jouni Smed
Mika Johnsson
Mikko Puranen
Timo Leipälä
Olli Nevalainen



Turku Centre for Computer Science
TUCS Technical Report No 196
August 1998
ISBN 952-12-0257-2
ISSN 1239-1891

Abstract

The arrangement of operations in a production line for mounting the surface components on a printed circuit board is discussed. The production program includes a wide range of different products, which causes frequent set-up operations. The overall efficiency of the production line depends heavily on how the printing operations are organized. Set-ups cause delays which can be cut down by selecting carefully the feeders for the components and by solving a suitable sequence for the products. We describe an integrated production management system for job grouping. The system utilizes approximative algorithms for minimizing the number of component switching instants. A discussion of the exact minimization by using mathematical 0/1 integer programming approach is also given. The revision of the production management system has had a major impact on the productivity, and an increase of ca. 58 percent in the number of component insertions per hour is observed.

This report is a revision of TUCS Technical Report No 139, which considered *batch selection problem*, whereas the present work is focused on *job grouping problem* in the same production environment. Furthermore, improved algorithms, a new mathematical model and a survey of relevant literature are presented here.

Keywords: printed circuit boards, group technology, approximative algorithms

TUCS Research Group
Algorithmics

Introduction

In flexible manufacturing systems (FMSs) [1] several different types of products are manufactured by the same production facilities. FMS reduces the machinery investments and widens the product range. However, it also provides us with many challenging production planning and management problems.

We present in this paper a production environment which comprises batches of different products, each demanding special set-ups, parts, tools and numerically controlled execution (NCX) codes. We assume that the set-up times caused by the product changes are non-negligible, and therefore we can decrease the *set-up costs* by sequencing the products suitably. In addition to the goal of minimizing the set-up costs, the production must meet *due dates*. The management wants to utilize the production facilities maximally and to cope with the short delivery times. At the same time it is preferable if the finished products do not pile up in internal storages.

We can control the finishing times—and in this way also the fulfillment of the due dates—by *scheduling* the jobs. In many cases the production planner is mainly concerned with the due dates, and the minimization of the set-up times may sometimes conflict with this goal. The construction of a feasible and, preferably, an efficient schedule is one of the most difficult tasks in production planning. It has been shown that the problem is too complicated to be solved accurately (even in theory) [2, 3]. Therefore, the problem is usually approached by the use of approximative algorithms.

In this paper we discuss the organization of an actual production plant. Furthermore, we concentrate on a single machine despite the fact that the machine is only a part of the production process. The reason for this simplification is the dominating role of this work phase compared to all other phases in the system. Our research originates from a *printed circuit board* (PCB) assembly plant (Teleste Corporation, Nousiainen, Finland) and aims at a solution for everyday use. The production line considered here is capable of manufacturing a large selection of different *jobs* (i.e., batches of PCBs). During the last two years the number of different PCB types has been over 350, and their respective annual production volumes vary from one to several thousand boards. Because the batch sizes are relatively small, the production is highly flexible and demands several set-ups daily.

Research work on PCB assembly has been done by many authors. Crama *et al.* [4] divide the problems of PCB production planning into five categories:

1. partitioning the set of board types into families,

2. partitioning the set of component locations for each board type,

3. determining the feeder set-up for each machine,

4. assigning a component placement sequence for each pair consisting of a machine and a board type, and

5. assigning a component retrieval plan for each pair of machine and board type.

Crama *et al.* focus on problems 2–5, whereas our focus is mainly on problem 1 because we concentrate on one line and one machine instead of multiple lines, but problems 3 (feeder optimization) and 4 (order optimization) are also taken carefully into account by our system.

Ammons *et al.* [5] categorize the strategies for set-up management as follows:

1. *Single set-up strategy* in which a group of machines is configured to produce a family of PCBs using a single set-up:

   (a) *Unique set-up strategy* in which the family contains only one product type (i.e., mass production).

   (b) *Family set-up strategy* in which the family comprises several product types.

2. *Multi set-up strategy* in which limited component staging capacity prohibits applying the single set-up strategy (see also [6]):

   (a) *Decompose and sequence* in which the family is divided into subfamilies which are then sequenced to minimize the incremental set-ups between subfamilies.

   (b) *Partition and repeat* in which the required components are partitioned into subsets restricted by machine capacity.

Our approach uses the multi set-up strategy because the number of components vastly exceeds the capacity of the machine. We will show later in this paper how to apply the decompose and sequence strategy to the actual production process.

Maimon and Shtub [7] present a classification of the approaches for sequencing PCBs and components on an automatic assembly line. This classification divides the problem in four categories according to the number of times each PCB is loaded (one or many) and the number of times each component is loaded (one or many). In our problem the PCBs are loaded only once, whereas the components can be loaded several times. Therefore, we

have three different strategies for the set-up: a static set-up which is identical for all products, a set-up which is identical for only a part of the products (family set-up), and every product has its own set-up. The first strategy is straightforward but, unfortunately, it is not applicable in our case because the total number of different component types exceeds the feeder capacity. The third strategy resembles the situation before the introduction of the system presented in this paper. The second strategy is a compromise: here we form standard set-ups for product groups (i.e., different products can use the same set-up). Note that in the first strategy we have one single family whereas in the third strategy each product forms a singular family (or product group). The difficulty with the family set-up strategy is that once the families have been assigned, the jobs should be sequenced on family basis, not independently. Nevertheless, this strategy forms the basis of the solution presented in this paper: *the products are divided into groups which are then sequenced by the production engineer.*

Another usual presumption is that the jobs are first sequenced and after that the set-up is optimized (cf. [8]). In our case optimizing the set-up for a given group is the most important task. Sequencing, which is done later, is left for the user to decide. Also, the usual optimization criterion reported in literature is minimizing the number of set-up operations (i.e., the total number of loaded components), whereas our goal is to *minimize the number of set-up occasions* (i.e., the number of instants the set-up operations occur). Tang and Denardo [9] present analyses for both minimizing the number of tool switches and minimizing the number of switching instants (or set-up occasions). In the first case we assume that each tool switch increases the processing time linearly. In the latter case we assume that the set-up of the tools for the next group is parallel to the processing of the previous groups and, therefore, we should minimize the number of job groups.

Crama and van de Klundert [10] define the loading strategy as a specification of the contents of the tool magazine at the beginning of the processing of each job and identify four basic scheduling problems:

1. *Tool switching* in which we try to find an input sequence and a loading strategy with a minimum total set-up time when it depends linearly on the number of switches.

2. *Loading problem* in which we try to find a loading strategy for a given part input sequence when the minimization criterion is the same as in case 1.

3. *Job grouping* in which we try to find an input sequence for the parts and a loading strategy for the tool magazine with a minimum total set-up

time when it depends linearly on the number of switching instants.

4. *Batch selection* in which we try to find the largest group of jobs that can be processed without tool switches.

Originally we considered the problem presented in this paper as an iterative batch selection problem (cf. [11]). However, because minimizing the switching instants (or the set-up occasions) turn out to be the most important criterion, our problem is a variant of the job grouping.

Carmon *et al.* [12] divide traditional approaches to reduce set-up times into two categories:

- reducing set-up frequency by enlarging the lot sizes, and
- *Group Technology* (GT).

In Group Technology, efficiences in manufacturing are realized by grouping similar tasks and dedicating equipment for performing these tasks. GT is often considered to be a part of medium range planning (cf. [7] for descriptions of long-, medium- and short-range—or strategical, tactical and operational—production planning in PCB assembly), but in our case the dynamic nature of the production environment enforces us to apply GT in the short-range planning. This feature is contrasted, in addition to those mentioned above, with the works by Brandeau and Billington [13], Hillier and Brandeau [14] and Carmon *et al.* [12]: In [14] the component set-up lasts for 6–12 months, whereas in our case the set-up changes several times weekly. In [13] the environment is similar with respect to a wide mix of boards and a low production volume for each board, but the line comprises semi-automated machines and each board is produced individually (i.e., there are no batches).

A Group Set-Up (GSU) method is presented in [12]. GSU works through four steps: set up the common components of the group, print them on the whole group, set up residual components of each PCB, and print them on each PCB separately. Despite the similarities, GSU differs from the strategy presented in this paper in that the common and residual components—or *standard set-up* and *custom set-up* components, as Ammons *et al.* refer to them in [5]—are printed simultaneously by the same machine. Because reloading is not allowed, we do not break the families when printing the residual or custom components, as in GSU.

Crama *et al.* [2, 10] give a solid theoretical background for the tool management problems and prove that job grouping is $\mathcal{NP}$-hard. They discuss the worst case of eight known batch selection algorithms and show their potential inefficiency. However, we must stress that their "job-sequencing and

tool loading" problem make few assumptions in the theoretical model, and this makes the problem somewhat different from ours. For example, it is assumed that each tool (or component) fits in one slot of the magazine; this is not the case in the environment described in this paper. Also, we must cope with due dates and some production-dependent relations of PCBs (e.g., PCB widths and two-sided boards).

The remainder of this paper is organized as follows. We start by describing the component printing line. This section contains some technical details which are needed in order to clarify the aspects of the problem in question. Next, the inefficiencies in the operation and our proposal for a revised production planning system is described. We also describe a new method for updating the control programs of the machine and strategies for filling the feeders. Next, we study what are the possibilities to improve the overall production by grouping the products (job grouping) and present a mathematical model and heuristic algorithms for solving this problem. After that the working system is described and an evaluation of the effects of the new system is given. The paper is closed by concluding remarks.

The Production Environment

Let us consider a typical assembly line for automatic component printing, see figure 1. The line comprises several successive work phases. An initially empty PCB passes first a *glue dispenser* which inserts a glue dot at each onsertion locus or draws adhesive paste over the whole board in order to fixate the electric components. The actual printing is done by two onsertion machines. The first one, *surface mounting device* (SMD), is adapted to fast operation and it is used for the majority of the component onsertions. Components which require specialized tools are onserted by a more flexible

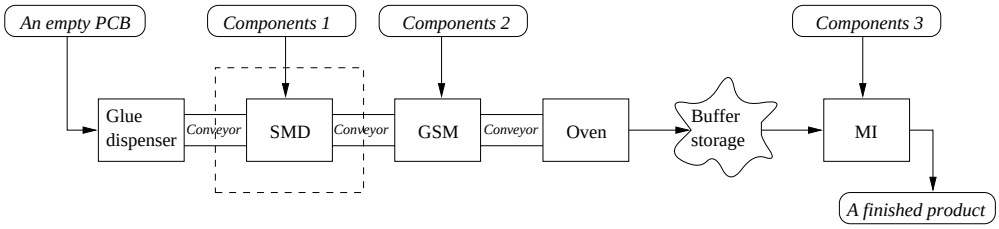


Figure 1: The production phases of the surface mounted onsertion. Legend: SMD = surface mounting device, GSM = general surface mounting device, MI = manual insertion

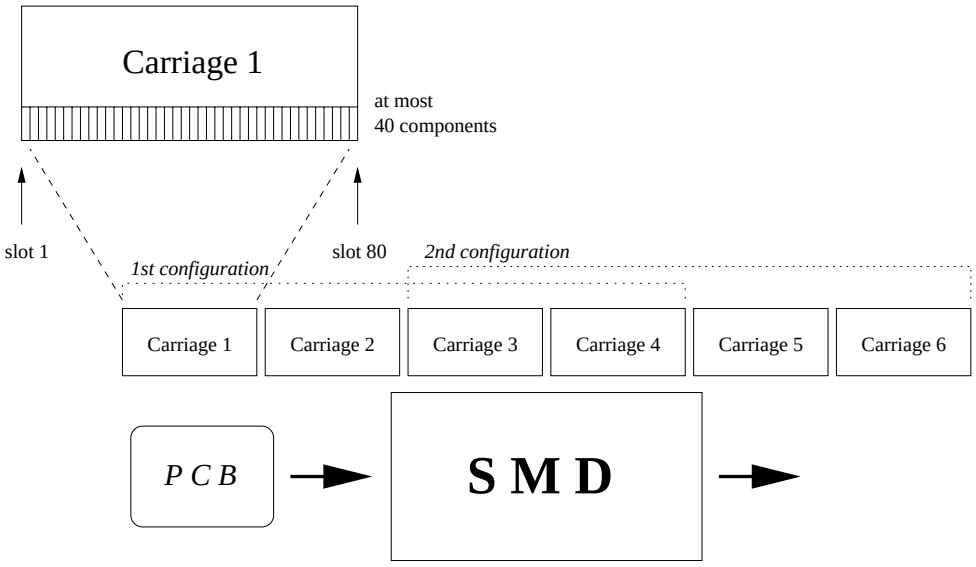


Figure 2: A schematic view of the SMD printing machine

but slower robot, a *general surface mounting* (GSM) machine. These two onsertion phases are followed by an *oven* which heats the PCBs in order to harden the glue/paste. After that the PCBs wait in a *buffer storage* and finally pass a *manual insertion* phase in which some large components are inserted and soldered.

If we look at the different jobs (PCB batches) which are processed on the line, we notice that their total number is very high but the amount of PCBs in a job is usually small. The daily production program includes typically 4–10 different *products* (PCB types). The set-up times form a significant part of the total production time—it can be as much as 50 percent. Therefore, our main objective is to minimize the set-up times by arranging the products efficiently. The due dates are managed by a two-level priority classification: products are either *urgent* or *non-urgent*. The last feature that affects the overall production time is that there are two different widths for the PCBs. Therefore, if the next PCB has a different width than the previous one, the width of the conveyor must be changed.

Because the set-ups and component printing of the SMD machine [15] consume most of the production time, it is the bottleneck of the whole production line. The difficult management of the machine in a *multi-model production* environment emanates from its flexibility. The machine gets the surface mounted components from six *carriage modules*, see figure 2. The following technical details should be taken into consideration:

1. Each carriage includes 80 linearly arranged *feeder slots* which contain the components. A component takes at least two slots giving thus each carriage a theoretical maximum capacity of 40 components, and 240 components for the whole machine. Because the sum of different component types in a PCB is significantly smaller than the capacity of the machine, we can quite freely choose an appropriate input organization.

2. The two outermost carriages can be separated from the unit while the machine is operating. The operator can carry out the set-up procedure for carriages 1 and 2 (or 5 and 6) while carriages 3–6 (or 1–4) are active in the component printing (configurations shown in figure 2). While the machine is working, the set-up of the midmost carriages 3 and 4 cannot be changed. Therefore, they should contain the most commonly used components.

The Structure of the New System

Before the system described in this paper was introduced, the existing organization of the production line needed some refinements. Previously carriages 2–4 were filled with the standard set-up components. The board was then printed with components from carriages 1–4 if the required components were contained by the standard set-up carriages 2–4 and the rest could be set up in carriage 1. Otherwise, the board was printed with standard set-up components from carriages 3–4 and custom set-up components from carriages 5–6. This method was, however, inefficient because of the following reasons:

1. The printing programs were laborious to update, which caused that the standard set-up components were seldomly changed. Therefore, the standard set-up gradually corrupted to contain components whose demand was not maximal any more. In addition, there was no efficient method for solving a new standard set-up.

2. Each product required a separate set-up because each printing program required a new setting for the custom set-up components. Furthermore, it was seldom possible to print one product during the set-up of another.

3. The order of the components in the feeders was not very efficient, which further reduced the productivity.

4. The printing order was poor.

The revised production planning system solves these inefficiences by introducing the following new features:

1. a method for choosing the standard set-up,

2. forming groups of jobs with the same feeder setting,

3. feeder optimization for product groups, and

4. using an efficient code generator that utilizes the current feeder set-up.

However, these features require a flexible and dynamic update process for the printing programs which is discussed in the next subsection. For the feeder and printing order optimization we use previously developed methods [16].

The new system uses a different partition of standard and custom set-up carriages where carriages 3 and 4 contain now a standard set of components, thus leaving on both ends symmetrically two carriages for the custom set-up. This layout enables continuous printing: while PCBs are being printed, the components of the next set-up can be placed on carriages which, at the time, remain idle on the other end of the machine.

Updating the Printing Programs

In the old system the printing program updating was problematic. Previously, when there were changes in the PCB or updates to the printing data, the changes were made directly to the NCX file and not to the CAD file. The NCX files (a file type used by Universal HSP 4790/91/92 and Sanyo TCM 1000/1100 SMD printing machines) contain the feeder set-ups for each PCB. The program which creates them has two inputs: a CAD file of the PCB and a file containing the standard feeder set-up. If the changes were made to the CAD file, the NCX file would have to be recreated by using the CIMBrigde-system [17], which takes a long time and is generally a laborious task. The minor changes made to the file continued to add up until it was no longer efficient to use the original source files.

Figure 3 presents a solution to the problem: The NCX files are now updated by using a new program which uses the old NCX file and a new feeder set-up file as an input and updates the NCX file accordingly. The updated NCX file still contains the changes made to the original NCX file. The revised update procedure enables us to introduce new feeder set-ups, which was a troublesome operation in the old system. This is required when we introduce the concept of product groups. These groups are dynamic, and therefore every time a product is manufactured, it requires an updated NCX file.

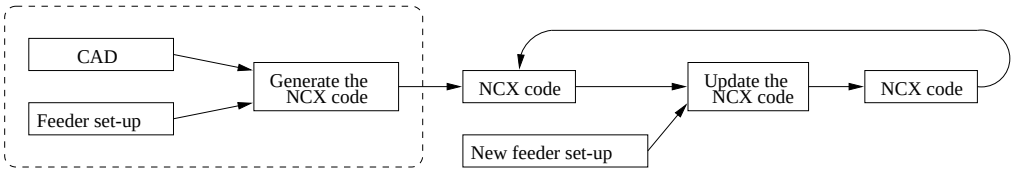


Figure 3: The revised update procedure

Choosing the Standard Set-up

By using the production data we have to decide which components should belong to the standard set-up (i.e., in carriages 3 and 4). In the current system we pick first the most frequently produced PCBs. From these PCBs we select 80 most frequently used components (which take the available 160 feeder slots), omitting wide components and components which are slow to print. On the whole, choosing a standard set-up has turned out to be a noncritical factor with respect to the overall operation of the plant.

Grouping

In this section we return to the second feature of the planning system: forming job groups with identical feeder set-ups. In the previous section we presented a strategy in which a part of the jobs has the same set-up. These jobs form a group [18, 7] and they are printed successively. In addition, there is no need for set-up operations between jobs belonging to the same group. If the printing of a group takes a longer time than the set-up of the next group, the group changing does not cause a delay and the PCBs can be printed continuously. Therefore, the products are classified into groups according to their *closeness* (i.e., the amount of mutual components). Products belonging to the same group will succeed each other (if possible). Thus, they use the same carriages and leave the other end of the carriage unit free for the next set-up.

Let us give an example of the grouping in which we want to process jobs A, B,..., J (figure 4). Jobs ABC belong to the same group and use a wide conveyor, while the narrow PCBs form groups DE, F and GHIJ. Job D is urgent. Figure 4b shows a possible sequence for the groups. The group DE is processed first due to the urgency of job D. The components belonging to the custom set-up of the group DE are located in carriages 1 and 2. The custom set-up of the group GHIJ uses carriages 5 and 6, the next group (F) again in carriages 1 and 2 and the last group (ABC) in carriages 5 and 6.

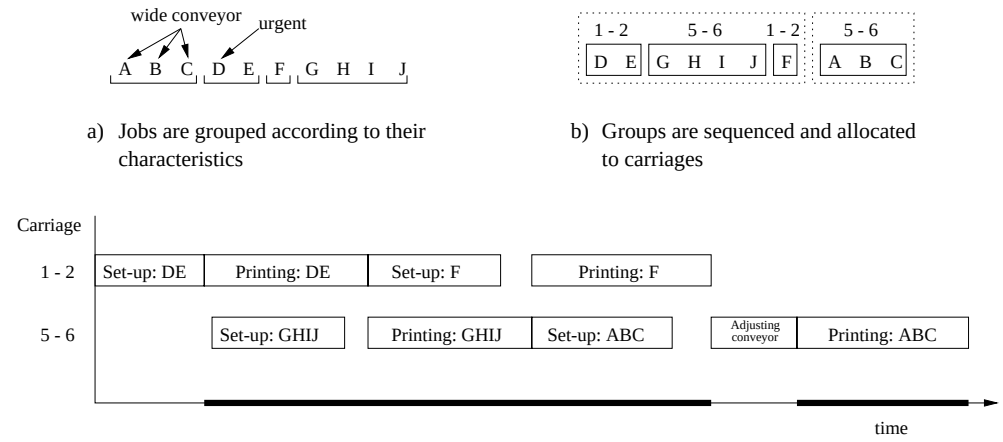


Figure 4: An example of job grouping

See figure 4c for an illustrative Gantt diagram of the processing.

The remainder of this section in divided as follows: First, we present a mathematical formulation for the grouping problem. Secondly, we concentrate on heuristic approaches. Finally, we give test results for these algorithms.

Mathematical Formulations for Grouping

We can formulate the job grouping problem as a mathematical optimization problem [19]. In this section we present a model for minimizing the number of groups or equivalently the number of set-up occasions (see [11] for a model which maximizes the size of the group). For this model we introduce the following parameters:

d_i the number of feeder slots needed by component i
$m:$ the number of different PCBs
$n:$ the number of different components
$k:$ the capacity of the carriage
$a_{ij}:$ the number of component i needed by board j

The set-up occasions are indexed with ℓ where $\ell = 1, \ldots, L$. Therefore, the carriage is loaded at most L times (estimated beforehand).

Let us now denote the decision variables

$$x_{i\ell} = \begin{cases} 1, & \text{if component } i \text{ is set up in the carriage in} \\ & \text{the occasion } \ell \\ 0, & \text{otherwise} \end{cases}$$

$$y_{j\ell} = \begin{cases} 1, & \text{if board } j \text{ is printed with set-up } \ell \\ 0, & \text{otherwise} \end{cases}$$

$$z_\ell = \begin{cases} 1, & \text{if set-up } \ell \text{ is required} \\ 0, & \text{otherwise} \end{cases}$$

Then we get the minimal number of groups by solving

$$1 \cdot z_1 + 2 \cdot z_2 + \cdots + L \cdot z_L = \min! \tag{1}$$

and

$$\begin{cases} x_{1\ell}d_1 + x_{2\ell}d_2 + \cdots + x_{n\ell}d_n \leq kz_\ell, \ell = 1, \ldots, L \\ y_{j\ell} \leq x_{i\ell}, \ell = 1, \ldots, L; i, j \ni \min\{1, a_{ij}\} = 1 \\ y_{j1} + y_{j2} + \cdots \ y_{jL} = 1, j = 1, \ldots, m \\ x_{i\ell} \in \{0, 1\}, y_{j\ell} \in \{0, 1\}, z_\ell\{0, 1\}. \end{cases} \tag{2}$$

This formulation has many equivalent solutions ($\tilde{\ell}!$ if $\tilde{\ell}$ is the optimal amount of set-up occasions), and therefore we can add the restriction

$$y_{1,\ell+1} + \cdots + y_{m,\ell+1} \leq y_{1\ell} + \cdots + y_{m\ell}, \ell = 1, \ldots, L-1. \tag{3}$$

In this case the size of the first group is the greatest, the size of the second group the second greatest etc. Note that the numbering does not restrict the scheduling of the job groups at the moment of actual production.

Due to the large size of the problem (1), (2), (3) we will give in the next section a number of approximative solution algorithms for the problem.

Heuristic Methods

Originally we considered five algorithms based on greedy selection which try to maximize one group at a time [11]. This problem is called the batch selection problem by Crama and van de Klundert [10], and its repeated application has commonly been used while solving the job grouping problem (see [10] for an analysis of eight batch selection heuristics). The algorithms presented in [11] differ in how they construct the next group. On a coarse level the methods work as follows:

- *Inclusion methods* add a new job (or job pair or job triplet) to the current group so that its inclusion causes a minimal increase in the number of the different components in the group. This procedure is repeated until the capacity of the feeders is exceeded. This method resembles Greedy Board algorithm described in [13].

- *Weighting method* adds the jobs in decreasing order of a weighted similarity measure.

- *Exclusion method* begins with a group containing all PCBs. After that it excludes the PCB which releases the greatest number of feeder slots, and the same is repeated until the components of the group fit in the carriages. This method resembles Stingy Component algorithm described in [13].

However, the maximal group size turns out be an unsatisfactory criterion because the groups tend to have a highly uneven distribution (i.e., there is one large group and several small ones). Therefore, we will next give a set of approximative algorithms which try to minimize the total number of groups. Thus, we abandon the greedy algorithm and try to solve the job grouping problem directly.

These algorithms begin with an initial solution given by a *clustering method* which is related to the clustering techniques used in other contexts, cf. PNN algorithm [20]. Among the several possible variations we consider the following method: Begin with forming singular groups so that each PCB forms a group of its own. Search for group pairs that can be merged into a single group which does not exceed the feeder capacity. Select the pair which gives the greatest benefit among all possible pairs (i.e., if the set R_i contains the components of the group i, and R_j the components of the group j, the pair (i, j) is selected so that $\Delta R = |R_i| + |R_j| - |R_i \cup R_j|$ is maximal). Merge the groups and repeat the merging procedure if possible.

Our algorithms begin with the initial result and then try to *further improve the grouping*. Improvements are likely since the heuristic merges only two groups at the time and, therefore, the original grouping is coarse and contains non-full groups. This free space can be covered with components which may make it possible to include a new PCB in the augmented group. We have two local search [21] operations to achieve this goal: *swap* and *merge*.

Swap operation examines all group pairs and swaps jobs (or PCBs) between the groups if it results a decrease in the feeder demand. In addition to the pair-wise swapping, we can try to move a job from one group to another. The objective is to decrease the total number of feeders required by the group. As a result from the move and swap operations, we can possibly apply the merge operation and, consequently, decrease the number of groups.

Merge operation incorporates one group to another group. Normally this operation does not succeed easily and, as a result, we try to merge two groups applying a selected rule. In this case we merge a group pair for which ΔC

is maximal *and* the excess of the capacity is less than a predetermined limit. The result of this operation may be—and most probably is—illegal: the new group violates the capacity constraint. The violation is then corrected by moving jobs from the group until the required set of components does not exceed the feeder capacity any more. The superfluous PCBs can be removed from the group by applying one of the following strategies:

1. Move the PCB that causes minimal increase of the feeders in the target group.

2. Move the PCB that releases most capacity in its source group.

3. Move the PCB for which the difference of the two above mentioned measures is minimal.

The problem in the first strategy is that it does not take into account the amount of released feeder slots in the source group, and in the extreme case no slots will be released. Obviously this is not desirable since our intention is to make room in the source group. It follows that the third strategy suits best for our requirements: it selects the source group and the PCB that maximize the amount of released slots and the target group that minimizes the increase of the feeder load.

In both swap and merge we have intentionally not restricted the possible operations to those yielding a feasible result (i.e., the grouping may still violate capacity constraints after the operations). This allows us to move to a different part of the solution space more freely than with a sequence of feasible groupings only.

The operations form the basis for the following five approximative algorithms for the PCB grouping.

HK1 (Clustering method) Construct an initial solution with m groups where each group is singular (i.e., consists of only one PCB). Merge group pairs in a greedy fashion so that ΔR is maximal and no feeder capacity constraints are violated. Iterate the merge operation as long as it is successful.

HK2 In this algorithm we begin with applying method HK1, and after that we try to improve the result with a combination of swap and merge operations which are continued until they do not provide us with an improved grouping. In the merge operation we select a group pair with the maximal ΔR value.

HK3 In this algorithm we begin with algorithm HK2 and after that try to improve the result with a local search heuristic which uses the swap operation. We decrease the number of groups of the initial solution by one and remove the superfluous group by allocating randomly the jobs of the smallest group to some other group. After this the grouping is likely to be unfeasible because some of the capacity constraints are violated. We solve this by moving jobs from one group to another and swapping jobs between two groups. Single job moves are used in the beginning, and when they are no more successful, they are switched to job swap operations. The quality of the grouping is evaluated with a *cost function* H. Minimizing only the number of groups is too coarse a criterion (i.e., it does not provide us with the indication whether the algorithm is moving to the right direction in solution space), and therefore the cost function is used to guide the search. This function sums the total number of the feeder slots $H = \sum_{j=1}^{L} h(j) + V \cdot C$, where $h(j)$ stands for the number of slots required by the group j, L the number of groups, V the number of slots exceeding the capacity and C is a penalty constant for the criteria. Hence, the aim is to minimize the value of H. Next, if the result is feasible, we record the solution, decrease the number of groups by one and restart the procedure. If the search was unsuccessful, the situation is again randomly initialized and the procedure restarted. This is repeated until a given time limit (in our tests 1 minute) has elapsed.

HKA1 In this algorithm the grouping order is controlled by defining a *group similarity measure* s_j for all groups j:

$$s_j = \sum_{i=0}^{k} \frac{i \cdot N_i}{k+1}$$

where k stands for the number of different components in the group j, and N_i is the number of components of the group j that are used in i other groups beside the group j. In other words, a group with a high similarity measure contains a lot of components which are used also in some other groups, and therefore its merging should not be too costly.

The algorithm sorts the groups into a decreasing order according to the similarity measures. The first group is then merged with some other group if possible; if not, the second group is merged with some other group etc. After the merge operation the similarity measures are updated and the groups sorted. The procedure is repeated until there is no improvement.

HKA2 Similar to HKA1 but the result is improved by applying method HK2 which performs the swap and merge operations.

Size	HK1	HK2	HK3	HKA1	HKA2	Best max	Overall best
10	1	1	1	1	1	1	1
15	3	3	3	3	3	3	3
20	3	3	3	4	3	3	3
25	4	4	3	4	4	4	3
30	4	4	3	5	4	4	3
35	4	4	4	5	5	4	4
40	7	5	5	6	5	6	5
45	5	4	4	6	4	4	4
50	5	5	5	7	5	6	5
60	7	6	6	8	6	7	6
70	8	8	8	8	7	8	7
80	7	7	7	8	8	7	7
90	9	9	9	10	9	10	9
100	9	9	9	11	8	10	8
125	12	10	10	13	11	12	10
150	12	11	11	14	11	13	11
175	14	12	12	15	12	13	12
200	15	13	13	17	13	16	13
225	15	15	15	18	14	16	14
250	18	15	15	19	16	17	15
341	20	16	16	23	19	20	16
Total	182	164	162	205	168	184	159
%	14.5	3.1	1.9	29.0	5.7	15.7	0.0

Table 1: The number of different groups produced by the solution

Test Results

In table 1 we present results for the improved heuristic methods presented in the previous section. The test problems were constructed by sampling random PCB types from the actual production data. We can conclude from these results that methods HK2, HK3 and HKA2 give quite similar results. HK3 turns out to be the best method giving results which on average are 1.9 percent from the optimum. In the best maximal grouping method (Best max) column the best result from the previous methods is duplicated (more detailed results appear in [11]). The improved methods were able to outdo the previous best in 15 cases out of 21 and by 15.7 percent.

The structure of the method HK3 is demanding and, consequently, it

Method	Average group size	Excess over optimum (%)
HK1	4.30	11.2
HK2	4.13	6.9
HK3	3.87	0.0
HKA1	4.77	23.3
HKA2	4.30	11.2
Best	3.87	0.0

Table 2: Results from 30 cases of problem size 30

requires more time than the other methods. The benefits of the method HKA1 are its speed and good efficiency throughout different problem sizes. However, all the methods are quick enough for problem sizes found in actual situations.

In table 2 a summary of results from another test set is presented. In this case each grouping problem comprised 30 randomly chosen PCBs (this problem size represents the average weekly production size). This was repeated 30 times for each heuristic method. Also, in this case method HK3 gives the best results. Additionally, we solved the optimal result for the same problems, and for all tested cases HK3 gave the optimal result. Method HKA1 gives the worst result, and its group size is over twenty percent from the best result. The running time for all heuristics, excluding HK3, is ca. 1–2 seconds; because of the time limit, method HK3 requires one minute of running time.

System Description

The system depicted here includes tools for grouping the jobs of the production plan and optimizing the set-up for each group [22]. When using the system, the production engineer can

- choose the products from a product list,
- choose which heuristic method is applied to form the groups,
- assign the standard set-up,
- assign the custom set-up for a given group and carriers (1–2 or 5–6),
- produce optimized NCX codes for a whole group or for some jobs within a group,

- compare two groups in order to discern mutual components and their respective locations,
- view the current groups by listing the jobs, the components or the feeder set-up,
- remove groups,
- move jobs between groups,
- get graphical and numerical information of the groups and the jobs,
- inspect whether a new job can be inserted to some existing group, and
- edit the component library.

In the current system (see figure 5) the production engineer can freely resequence the groups and the jobs within a group. The latter does not affect the set-up times since no set-up is needed for jobs in the same group. The due date, the batch size, the conveyor width and the availability of required components are the most important factors when the production engineer is constructing a feasible schedule.

Experiences

In order to demonstrate the performance of the new system we present an actual production plan for a period of one week, see table 3. The production schedule is created by using both the old system and the new system. We can calculate the number of required component changes, component change operations (i.e., set-ups) and the expected printing time for each board by using a simulator presented in [16]. From this data we can calculate the total time required to produce the whole production program in both systems.

The benefits of the new system are obvious (see table 4). In the old system we have to do 21 set-ups, one for each product, while in the new system products are divided into three groups, each requiring a single set-up. Also, the printing time is vastly reduced; in this case the reduction is ca. 18 percent. It must be noted that we have included only the times which differ in the new and the old system. However, there is a lot of processing (program loading, line changing, coping with breakdowns and other problems) which is not affected by the change of the system, and therefore this analysis is not complete. We believe that the decreased number of set-ups will increase the production much more than these estimates suggest, the reason being that

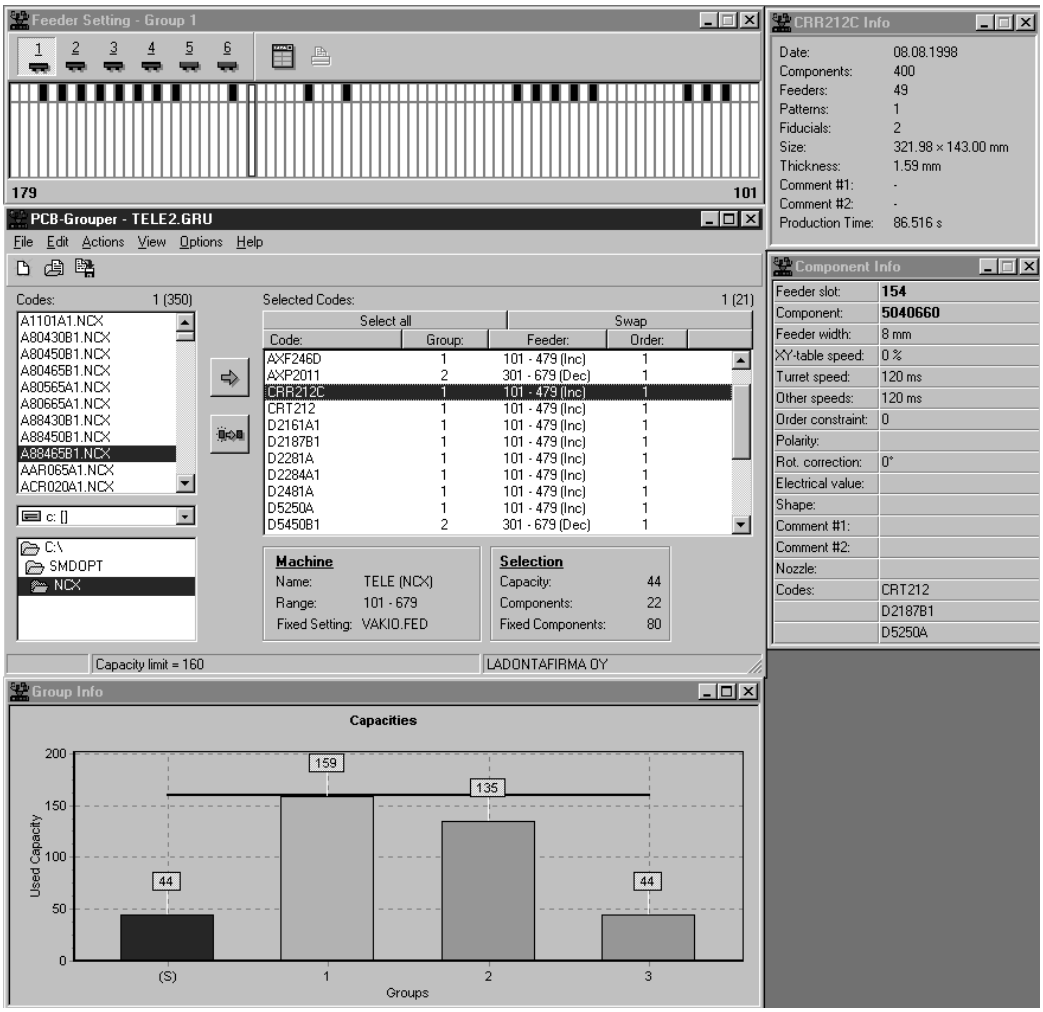


Figure 5: Main window of the grouping system and information windows for feeders, boards, components and groups

PCB	pcs.	Old	New	Improvement	%	Group
D2284A1	200	48.61	32.64	15.97	32.9	1
CRT212	30	96.62	70.90	25.71	26.6	1
D5250A	100	55.06	40.73	14.34	26.0	1
D5450B1	210	42.87	34.13	8.75	20.4	2
CRR212C	35	103.16	83.70	19.47	18.9	1
D2187B1	31	33.15	27.05	6.09	18.4	1
D2161A1	79	31.77	26.21	5.56	17.5	1
D2481A	100	37.42	30.92	6.50	17.4	1
D2281A	200	38.95	32.45	6.50	16.7	1
DXA821RE	52	37.52	31.78	5.74	15.3	1
S2203	30	47.46	41.20	6.26	13.2	1
M3151EC2	30	105.90	93.49	12.41	11.7	3
AXP2011	50	39.19	35.10	4.09	10.4	2
MHE6107	47	78.28	70.47	7.80	10.0	2
DXO802	200	37.10	33.48	3.62	9.8	1
A80430B1	30	62.06	57.42	4.64	7.5	2
DTU082B1	31	30.78	28.73	2.06	6.7	1
AXF246D	30	22.52	21.96	0.56	2.5	1
AXA860	56	36.73	36.73	0.00	0.0	1
M3153CC2	10	104.30	104.53	−0.24	−0.2	1
DTU121B1	31	27.63	29.70	−2.07	−7.5	1
Total		1,117.07	963.31	153.76	13.8	

Table 3: An analysis of a sample schedule for a period of one week. Legend: Old = production time in the old system (min), New = production time in the revised system, Improvement = difference between Old and New, % = difference in percents, Group = group to which the product belongs

when there are only few set-ups, the whole organization of other tasks works more efficiently because of less interruptions.

The new system has been in production use since May 1997. The manufacturer has collected statistical data from the production. We compared the first ten-week period after the change with the preceding twenty-week period and observed the following improvements:

- The average number of component placements per hour (of total time) has increased by 57.6 percent.

- The average number of component placements per hour (of the actual printing time) has increased by 16 percent which is in accord with the

	Number of set-up occasions	Number of feeder changes	Total set-up time	Total onsertion time	Total production time
Old	21	294	30,240 s	72,868 s	103,108 s
New	3	246	16,560 s	60,067 s	76,627 s
Improvement	18	48	13,680 s	12,801 s	26,481 s
%	85.7	16.3	45.2	17.6	25.7

Table 4: Comparing the old and new system

results of table 3. The main reason for this is the improvements in printing and feeder optimization (cf. [16]).

- The average number of completed jobs in a week has increased from 22 to 28.
- The average time to change from one job to another (including machine set-up and other delays) on the whole line has decreased from an hour and a half to one hour (35.5 percent).

We realize that there has been minor changes in the product assortment during the test period. The average size of a batch has slightly increased which causes that the number of placements per hour (of total time) given here is somewhat optimistic. On the other hand, the number of completed jobs per week suffers from the growing batch sizes. Also, a slight improvement is due to purchasing new feeder reels which speeds up the set-up.

Concluding Remarks

We discussed the production planning in a flexible manufacturing line. The situation was abstracted as the management of a single machine. Even with this simplification we were confronted by a number of hard optimization problems. We solved the grouping problem and the code generation problem separately. These solutions formed the basis for the new system which introduced new ways to solve the scheduling problems in the production plant. The system described in this paper is in daily use and has decreased the average change time of a job by 35 percent. The number of component placements per hour has increased by 58 percent due to the new system and code optimization.

There remains yet a number of open research questions we would like to solve, for example, how to sequence the groups efficiently. The division

between the SMD and GSM machines should be reconsidered in order to gain a better balance for the load of the whole production line. We presented algorithms which group the products heuristically. The algorithms use the same framework: construction of initial groups and improvement of the grouping. We used the clustering technique for the initial grouping and local search methods for improving. Because there are several conflicting goals when forming the groups, we are currently developing new objective functions which are based on the fuzzy multiple criteria optimization.

Acknowledgement

The authors wish to thank Mr. Jouni Lehtinen from Teleste Corporation for his time and co-operation.

References

[1] A. Kusiak. Application of operational research models and techniques in flexible manufacturing systems. *European Journal of Operational Research*, 24:336–45, 1986.

[2] Y. Crama, A. W. J. Koleen, A. G. Oerlemans, and F. C. R. Spieksma. Minimizing the number of tool switches on a flexible machine. *International Journal of Flexible Manufacturing Systems*, 6:33–54, 1994.

[3] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-completeness*. Freeman, 1979.

[4] Y. Crama, O. E. Flippo, J. van de Klundert, and F. C. R. Spieksma. The assembly of printed circuit boards: A case with multiple machines and multiple board types. Forthcoming in *European Journal of Operational Research*, 1997.

[5] J. C. Ammons, M. Carlyle, L. Cranmer, G. DePuy, K. Ellis, L. F. McGinnis, C. A. Tovey, and H. Xu. Component allocation to balance workload in printed circuit card assembly systems. *IIE Transactions*, 29(4):265–75, 1997.

[6] R. G. Askin, M. Dror, and A. J. Vakharia. Printed circuit board family grouping and component allocation for a multimachine, open-shop assembly cell. *Naval Research Logistics*, 41:587–608, 1994.

[7] O. Maimon and A. Shutb. Grouping methods for printed circuit boards. *International Journal of Production Research*, 29(7):1370–90, 1991.

[8] J. F. Bard, R. W. Clayton, and T. A. Feo. Machine setup and component placement in printed circuit board assembly. *International Journal of Flexible Manufacturing Systems*, 6:5–31, 1994.

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

[10] Y. Crama and J. van de Klundert. The approximability of tool management problems. Technical Report rm96034, Maastricht Economic Research School on Technology and Organisation, 1996.

[11] J. Smed, M. Johnsson, M. Puranen, J. Lehtinen, T. Leipälä, and O. Nevalainen. On the work process organization of surface mounted component printing. Technical Report 139, Turku Center for Computer Science, 1997.

[12] T. F. Carmon, O. Z. Maimon, and E. M. Dar-el. Group set-up for printed circuit board assembly. *International Journal of Production Research*, 27(10):1795–810, 1989.

[13] M. L. Brandeau and C. A. Billington. Design of manufacturing cells: Operation assignment in printed circuit board manufacturing. *Journal of Intelligent Manufacturing*, 2:95–106, 1991.

[14] M. S. Hillier and M. L. Brandeau. Optimal component assignment and board grouping in printed circuit board manufacturing. Forthcoming in *Operations Research*, 1997.

[15] Universal Instruments Corporation. *Model 4792 Machine Operations Manual*, 2nd edition, 1996.

[16] M. Johnsson, T. Leipälä, and O. Nevalainen. Optimizing turret based SMD machines. In preparation, 1998.

[17] Mitron Corporation. *Line Scheduler User Manual Version 3.2*, 1995.

[18] K. R. Kumar, A. Kusiak, and A. Vanelli. Grouping of parts and components in flexible manufacturing systems. *European Journal of Operational Research*, 24:387–97, 1986.

[19] G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1988.

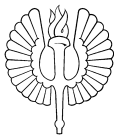
[20] A. Jain and R. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Englewoods Cliffs, New Jersey, 1988.

[21] C. R. Reeves. *Modern Heuristic Techniques for Combinatorial Problems.* McGraw-Hill, 1995.

[22] M. Johnsson. User manual for grouping software. Technical Report M-97-1, University of Turku, 1997.

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

http://www.tucs.abo.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**