Job Grouping with Minimum Setup in PCB assembly

KARI SALONEN1, JOUNI SMED1, MIKA JOHNSSON2, and OLLI NEVALAINEN1
1Turku Centre for Computer Science (TUCS)
and Department of Information Technology
University of Turku, FINLAND
2Valor Computerized Systems (Finland)
Ruukinkatu 1, FIN-20520 Turku, FINLAND

Abstract: We consider the machine setup problem of printed circuit board (PCB) assembly as a combination of a job grouping problem and a minimum setup problem. We formulate the problem as a MIP-model, where the objective is to minimize the weighted sum of the number of setup occasions and the total number of component feeder changes. We also present and evaluate hybrid algorithms based on both grouping and minimum setup heuristics. The best results are achieved by a method which uses both these strategies simultaneously.

Key-Words: Job grouping, Minimum feeder setup, Flexible manufacturing, Printed circuit board, Setup strategy, Heuristics

1 Introduction

Electronic industry uses automated systems to manufacture its products. The central part of an electronic product, printed circuit board (PCB) with its electronic components, is manufactured in assembly lines. A line can assemble components on multiple types of PCBs by one or several high-speed machines.

We examine the single machine setup problem in a high-mix low-volume environment where the total number of different component types of all PCB types exceeds machine’s component feeder capacity, and feeder setup operations are needed between jobs. This environment is typical in small PCB manufacturers who tend to have only few assembly lines and a large variety of PCBs to produce. More pressure is still added to setup operations, if prototypes are assembled in the same line. Efficient use of bottleneck machines is then one of the key aspects in production control.

We consider in this work group setup and minimum setup strategies [10] for organizing the setup operations. These two seem to be the most popular strategies in high-mix low-volume production environments in the single machine case.

Group setup strategy forms job groups of similar PCBs so that component setups are incurred between groups, only. Hence, any board type in the group can be produced without changing the component setup, which is only required when switching from one group to another. A natural objective is to minimize the number of groups. Numerous researches have studied the group setup strategy, see e.g. [12, 5, 2, 4, 13, 8].

Minimum setup strategy attempts to sequence the boards and determine component-feeder assignments to minimize the total component changeover time. The problem can be formulated as a special traveling salesman problem (TSP) and it can be solved heuristically. For solution heuristics, see e.g. [1, 15, 6, 7, 11].

It is common that a single component feeder can be changed in 1-5 minutes but it may take, for instance, 15-25 minutes to prepare the machine for the component setup operations, because it requires extra manual work by the personnel. Therefore, we have two different setups: a component setup comprises the operations to replace one component feeder with another, and a setup occasion takes place when the line is interrupted for one or more feeder changes. The cost function of the setup operations for a set of PCB assembly jobs on a single machine is then:

\[ C(y, z) = Ry + Sz, \]

where \( R \) and \( S \) are constant time factors for the number of setup occasions \( y \) and for the number of component changes \( z \). The aim is therefore to minimize (1).

The algorithms given in this paper are based on a method presented in [14]. In the original method, after grouping the PCBs each group is considered as a super-PCB. They are then used as an input for a minimum setup algorithm, which arranges the super-PCBs
so that the feeder changes between the super-PCBs are minimized.

2 Mathematical formulation

We formulate the hybrid machine setup problem, called here the Job Grouping with Minimal Feeder Setup (JGMFS) problem, as a MIP model. We follow Tang and Denardo’s [15] mathematical model (translated into PCB environment by Barnea and Sipper [1]), where the objective is to minimize the number of tool switches (component feeder changes). In addition, our formulation takes into account the number of setup occasions (groups).

Let \( N \) be the number of batches (jobs) of different PCB types. The capacity of the component feeders is \( C \) feeder slots and the total number of different components of all PCBs is \( M \). To simplify, we suppose that all feeder (reels) demand one feeder slot. Let \( A = \{ a_{ji} \} \) be a job-component matrix of the size \( N \times M \), where the element \( a_{ji} \) is 1, if at least one component \( i \) is required for the PCB type \( j \); otherwise, it is 0. The components required by the PCB type \( j \) are given by a \( 1 \times M \)-dimensional row vector \( A_j \). Let \( e(A_j)^T \geq C \) for each \( j \), where \( e \) is a \( 1 \times M \) row vector of 1’s, but \( C < M \); that is, each PCB type can be processed with a single setup of components but the components of all PCBs do not fit the feeders simultaneously. To fix the sequence of PCBs, we define a variable \( x_{jn} \), which is 1 if PCB \( j \) is \( n^{th} \) in the sequence, otherwise 0. Let \( W_{nj} \) be an \( 1 \times M \)-dimensional row vector that determines the components placed to the feeders at the instant \( n \) (\( 1 \leq n \leq N \)). In the beginning of the process there are no components in the machine (i.e., the elements of \( W_{n0} \) are all 0’s). The element \( w_{ijn} \) is 1 if the feeder for the component \( i \) is on some feeder slot of the machine at instant \( n \), otherwise it is 0. Let \( P_{jn} \) be an \( 1 \times M \)-dimensional row vector, where \( p_{ijn} > 0 \), if \( w_{ijn} = 1 \) and \( w_{ijn-1} = 0 \); otherwise, it is 0. Thus, \( P_{jn} \) tells the components to be introduced in the \( n^{th} \) product. To get the number of setup occasions (groups), we introduce a decision variable \( y_n \), which is 1 if at least one of the elements in \( P_{jn} \) is greater than zero; otherwise, \( y_n \) is 0. JGMFS is then stated as follows:

\[
\text{(2)} \quad \text{MIN} \sum_{n=1}^{N} (Ry_n + SeP_{jn}^T)
\]

subject to

\[
(3) \quad p_{ijn} \geq W_{nj} - W_{n-1} \quad \forall n = 1, 2, \ldots, N
\]

\[
(4) \quad eP_{jn}^T \leq Cy_n \quad \forall n = 1, 2, \ldots, N
\]

\[
(5) \quad x_{jn} \cdot A_j \leq W_{nj} \quad \forall n, j = 1, 2, \ldots, N
\]

\[
(6) \quad eW_{nj}^T \leq C \quad \forall n = 1, 2, \ldots, N
\]

\[
(7) \quad \sum_{n=1}^{N} x_{jn} = 1 \quad \forall j = 1, 2, \ldots, N
\]

\[
(8) \quad \sum_{n=1}^{N} x_{jn} = 1 \quad \forall n = 1, 2, \ldots, N
\]

\[
(9) \quad eW_{nj}^T = 0
\]

\[
(10) \quad p_{ijn} \geq 0 \quad \forall n = 1, 2, \ldots, N
\]

\[
(11) \quad w_{ijn}, y_n \in \{0, 1\} \quad \forall n = 1, 2, \ldots, N \quad \forall i = 1, 2, \ldots, M
\]

The objective function (2) minimizes the weighted sum of the number of setup occasions and the number of component feeder changes. Restrictions (3) determine component feeder changes for the \( n^{th} \) PCB in the sequence. The operators “\( \geq \)” and “\( \leq \)” in (3), (5) and (10) stand for the comparisons between the corresponding vector elements. Restrictions (4) ensure that a setup occasion happens every time when there are one or several component replacements when moving to a new job. Restrictions (5) ensure that all the components required for PCB \( j \) are on the machine at the instant \( n \). Restrictions (6) ensure that the number of allocated component feeders does not exceed the capacity. Restrictions (7) and (8) indicate that each PCB is processed exactly once and at each instant there is exactly one PCB under processing. The formulation generates \((N+1) \times (N+M)\) 0/1 -variables and \(N \times M\) real valued variables.

The problem (2)-(11) is interesting in many respects. By setting \( R > 0 \) and \( S = 0 \) we have a common job grouping problem. By setting \( R = 0 \) and \( S > 0 \) we have a tool switching problem where the objective is to minimize the number of tool switches (component feeder changes). However, joining these two objectives (i.e., \( R > 0 \) and \( S > 0 \)) gives us a still more realistic model of the machine setup problem. The number of setup occasions and the total number of feeder changes are both considered in this problem. Knowing that the hybrid problem is a combination of two NP-hard problems [3, 15], we can solve it optimally for small problem instances only.
3 Hybrid algorithms

Since the optimal solution of our MIP formulation for JGMFS is hard to find for problems of realistic size, we need to develop efficient heuristics.

3.1 Previous algorithms

We implemented in [14] three grouping algorithms, four minimum setup algorithms and a hybrid algorithm to the JGMFS-problem. We next recall the ideas of GSA1, MSAGenius, and GMSA1.

Algorithm GSA1 applies hierarchical clustering (HC) [10]. The algorithm uses Jaccard’s similarity coefficient \( s_{ij} = \frac{|E_i \cap E_j|}{|E_i \cup E_j|} \), where the set \( E_i \) (\( E_j \)) denotes the components of the board \( i \) (\( j \)). The algorithm starts by forming a single group from each individual PCB and then merges group pairs with the highest \( s_{ij} \). The process is iterated as long as possible.

As a post-processing phase, the number of groups is then reduced by move and swap operations and by applying a special localSearch heuristic, which allows the capacity constraint to be temporarily violated in order to escape a local minimum. Algorithm GSA1 can be described at a coarse level as follows:

**Input:** pcbs, a set of PCBs

**Output:** groups, a set of PCB-groups

**function** GSA1(\( PCB_{set} \) pcbs)

return localSearch(swap(move(HC1(pcbs))))

Algorithm MSAGenius (see [7]) starts by forming a complete weighted graph where the PCBs are nodes and the weight of the arc between node \( i \) and \( j \) is calculated from the expression \( |E_i \cup E_j| - |E_i \cap E_j| \) (giving the number of non-mutual components). The algorithms repeat the next four steps \( N \) times by taking each node as a starting node:

1. Solve heuristically an open TSP problem.
2. Improve the solution with a 2-opt heuristic.
3. Use keep tool needed soonest (KTNS [15]) method to assign components to feeders when visiting the nodes in the sequence given in step 2.
4. Evaluate the solution by function (1); store if better than current best.

Algorithm GMSA1 starts by forming PCB groups with GSA1 and then it sequences the groups with MSAGenius.

3.2 New algorithms

Algorithm GMSA2 starts the grouping of PCBs like GSA1, but it stops clustering when the similarities between the groups drop below a given limit. After that, the groups are sequenced and the feeders are assigned by MSAGenius. Note that when sequencing the groups formed by GSA1, it is possible that the number of the groups can still decrease. This happens if there are no feeder changes between two consecutive groups after sequencing. The following inequality is used as the similarity limit:

\[ 10 \times \max(s_{ij}) \times R > 3 \times S \]

where \( \max(s_{ij}) \) stands for the highest similarity coefficient among the feasible merging groups. The constants (10, 3) have been selected as a basis of test runs. The inequality implies that a small value of the ratio \( R/S \) requires a large similarity \( s_{ij} \) in order that the groups \( i \) and \( j \) will be merged. The problem then becomes a minimum setup problem and therefore one should leave more power for MSAGenius to arrange the PCBs. If \( R/S \) is large, the similarity limit should be lower, because it is now more important to minimize the number of groups. A pseudocode for GMSA2 is as follows:

**Input:** pcbs, a set of PCBs

**Output:** feeders, sequenced sets of component feeders

**function** GMSA2(\( PCB_{set} \) pcbs)

return MSAGenius(
    makeSuperPCBs(GSA1(pcbs))
)

**function** GSA1(pcbs)

return HC2(pcbs, \( R, S \))

**function** HC2(\( PCB_{set} \) pcbs, \( R, S \))

\( GROUP_{set} \) groups

for every \( pcb_i \) \( \in \) pcbs

change \( pcb_i \) to \( group_i \)

add \( group_i \) to groups

endfor

calculate similarity coefficient \( s_{ij} \)

for every \( groupPair_{i,j} \) \( \in \) groups

while ((feasible merging is possible) and

(\( 10 \times \max(s_{ij}) \times R > 3 \times S \))

merge the pair with the \( \max(s_{ij}) \) among the feasible merging groups

update similarity coefficient \( s_{ij} \)

for every \( groupPair_{i,j} \) \( \in \) groups

endwhile

return groups
Algorithm GMSA3 forms groups as in GSA1 (without the post-processing phase), but each time after merging two groups, it calls MSAGenius, evaluates the cost function, and saves improved solutions. The advantage of this method is that it searches results more globally than GMSA2. Algorithm GMSA3 for solving the JGMFS-problem:

**Input:** pcbs, a set of PCBs  
**Output:** feeders, sequenced sets of component feeders

```plaintext
function GMSA3(PCB set pcbs)
    return GSA1(pcbs)
endfunction

function GSA1(pcbs)
    return HC3(pcbs)
endfunction

function HC3(PCB set pcbs)
    PCB set pcbs2  
    GROUP set groups  
    Cost cost1=∞  
    Cost cost2  
    FeederAssignment set fa1  
    FeederAssignment set fa2  
    for every pcb i ∈ pcbs  
        change pcb i to group i  
        add group i to groups  
    endfor  
    while (feasible merging is possible)  
        calculate similarity coefficient s_{ij}  
        for every groupPair i,j ∈ groups  
            merge the pair with the max(s_{ij}) among  
            the feasible merging groups  
            pcbs2=makeSuperPCBs(groups)  
            fa2=MSAGenius(pcbs2)  
            cost2=costFunction(fa2)  
            if(cost2<cost1)  
                fa1=fa2  
                cost1=cost2  
            endif  
    endwhile  
    return fa1
endfunction
```

4 Computational experiments

We performed tests with GMSA1, GMSA2, and GMSA3 on two different datasets. The number of different PCBs in the dataset 1 and 2 are 20 and 40, respectively. We have repeated the experiment for each dataset 100 times using different PCB sets drawn randomly from an production program of a high-mix low-volume producer. Each PCB type contains 5-80 different component types. We experimented with the settings S = 1 and R = 0, 5, 10, and 20 and with the machine feeder capacity to 80 and 120. The average results of the experiments are shown in Table 1.

The results indicate that GMSA3 yields the overall best results. Only, when R = 20 GMSA1 gives sometimes better results than GMSA3. The reason for this is that GMSA3 uses GSA1 without the post-processing phase and therefore, the number of groups is less for GMSA1 than GMSA3.

Interestingly, when R = 0 (i.e., we are solving a pure minimum setup problem) GMSA3 is even better for our data than MSAGenius. Note especially the case, when the number of different PCBs is 40 and the capacity of the machine is 80; MSAGenius then gives 299.5 and GMSA3 295.4 component chances on average. The results of this case were compared using a paired t-test (two-sided). The test indicated that the difference of these results is statistically highly significant (p-value < 0.001). When the number of the PCBs is 20 and the capacity 80 (R = 0), the difference of the results is significant (p-value < 0.01).

For the capacity 120 (R = 0) the results of MSAGenius and GMSA3 are identical except for the case when the number of the PCBs is 40. Then the difference of the results is statistically significant (p-value < 0.01). A reason for GMSA3’s superiority over MSAGenius might be the following; when we form clusters some of the most similar PCBs before they are sequenced by MSAGenius, the problem size “decreases” and this helps us to get lower cost function values.

When comparing the results of GMSA1 and GMSA2, we notice that GMSA2 gives clearly better results for R = 0. On the other hand, when R = 20 GMSA1 outperforms GMSA2 with small margin. The reason for this is that GMSA1 always first minimizes the number groups despite the values R and S and then it sequences the groups by MSAGenius, whereas GMSA2 utilizes the values of R and S to leave more power for MSAGenius. This is beneficial especially if the problem is close to the minimum setup problem.
### Table 1: A summary of test runs with MSAGenius, GMSA1, GMSA2, and GMSA3. c- and cc-fields give the average number of different groups and feeder changes, respectively. cost-field gives the value of Formula(1).

<table>
<thead>
<tr>
<th>alg. capacity=80</th>
<th>R=0, S=1</th>
<th>R=5, S=1</th>
<th>R=10, S=1</th>
<th>R=20, S=1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>20 PCBs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSAGenius</td>
<td>16.1</td>
<td>212.0</td>
<td>212.0</td>
<td>10.4</td>
</tr>
<tr>
<td>GMSA1</td>
<td>5.0</td>
<td>223.3</td>
<td>223.3</td>
<td>5.0</td>
</tr>
<tr>
<td>GMSA2</td>
<td>15.8</td>
<td>212.1</td>
<td>212.1</td>
<td>5.5</td>
</tr>
<tr>
<td>GMSA3</td>
<td>14.9</td>
<td>211.5</td>
<td>211.5</td>
<td>5.5</td>
</tr>
<tr>
<td><strong>40 PCBs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSAGenius</td>
<td>32.9</td>
<td>299.5</td>
<td>299.5</td>
<td>23.9</td>
</tr>
<tr>
<td>GMSA1</td>
<td>8.4</td>
<td>332.1</td>
<td>332.1</td>
<td>8.4</td>
</tr>
<tr>
<td>GMSA2</td>
<td>34.5</td>
<td>299.2</td>
<td>299.2</td>
<td>9.0</td>
</tr>
<tr>
<td>GMSA3</td>
<td>30.0</td>
<td>295.4</td>
<td>295.4</td>
<td>9.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>alg. capacity=120</th>
<th>R=0, S=1</th>
<th>R=5, S=1</th>
<th>R=10, S=1</th>
<th>R=20, S=1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>20 PCBs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSAGenius</td>
<td>11.0</td>
<td>207.7</td>
<td>207.7</td>
<td>7.5</td>
</tr>
<tr>
<td>GMSA1</td>
<td>2.8</td>
<td>209.1</td>
<td>209.1</td>
<td>2.8</td>
</tr>
<tr>
<td>GMSA2</td>
<td>11.0</td>
<td>207.7</td>
<td>207.7</td>
<td>2.9</td>
</tr>
<tr>
<td>GMSA3</td>
<td>10.6</td>
<td>207.7</td>
<td>207.7</td>
<td>2.9</td>
</tr>
<tr>
<td><strong>40 PCBs</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSAGenius</td>
<td>28.6</td>
<td>263.7</td>
<td>263.7</td>
<td>18.9</td>
</tr>
<tr>
<td>GMSA1</td>
<td>4.2</td>
<td>280.8</td>
<td>280.8</td>
<td>4.2</td>
</tr>
<tr>
<td>GMSA2</td>
<td>28.6</td>
<td>263.5</td>
<td>263.5</td>
<td>4.7</td>
</tr>
<tr>
<td>GMSA3</td>
<td>27.7</td>
<td>263.2</td>
<td>263.2</td>
<td>4.7</td>
</tr>
</tbody>
</table>

5 Concluding remarks

We formulated a combined job grouping problem and minimum feeder setup problem in PCB assembly for a single machine case. The objective was to reduce the weighted sum of the number of setup occasions and the number of component feeder changes. This hybridization model simulates the real-world production planning situation where both of these factors should be considered. We presented two new hybrid algorithms (GMSA2, GMSA3) based on efficient grouping and minimum setup heuristics. Our practical tests indicated that the new algorithms can improve the solutions of the previous algorithms. The hybrid algorithm GMSA3 yields the overall best results. The results of GMSA2 are not far from GMSA3, whereas GMSA1 works satisfactory only if R/S is greater than 5. An interesting question of further research is the effect of job grouping and minimum setup on the assembly time. One should here compare the JGMFS strategy to that of performing unique setup for each single board and to the strategy of doing some partial rearrangements for the most frequently used component feeders. This kind of comparison could reveal problem dependent critical batch sizes for which one should abandon the JGMFS and switch to more careful optimization of the feeder allocation.
References


