The assembly of printed circuit boards: a case with multiple machines and multiple board types

Yves Crama[1]
Olaf E. Flippo[2]
Joris van de Klundert[2]
Frits C.R. Spieksma[3]

December 22, 1998

[1]Ecole d'Administration des Affaires, Université de Liège, Boulevard du Rectorat 7 (B31), 4000 Liège, Belgium.

[2]Department of Quantitative Economics, University of Limburg, P.O. Box 616, 6200 MD Maastricht, The Netherlands.

[3]Department of Mathematics, University of Limburg, P.O. Box 616, 6200 MD Maastricht, The Netherlands.

Abstract

In this paper a typical situation arising in the assembly of printed circuit boards is investigated. The planning problem we face is how to assemble boards of different types using a single line of placement machines. From a practical viewpoint, the multiplicity of board types adds significantly to the complexity of the problem, which is already very hard to solve in the case of a single board type. In addition, relatively few studies deal with the multiple board type case. We propose a solution procedure based on a hierarchical decomposition of the planning problem. An important subproblem in this decomposition is the so-called *feeder rack assignment problem.* By taking into account as much as possible the individual board type characteristics (as well as the machine characteristics) we heuristically solve this problem. The remaining subproblems are solved using constructive heuristics and local search methods. The solution procedure is tested on real-life instances. It turns out that, in terms of the makespan, we can substantially improve the current solutions.

Keywords: heuristics, PCB-assembly, feeder rack assignment problem.

1 Introduction

The assembly of printed circuit boards (PCBs) is in general a complicated task. Sophisticated machines must perform intricate operations, involving different kinds of tools and various components, in order to assemble a board. Numerous constraints and conflicting objectives interfere to create a challenging planning problem. Further, the competition faced by a PCB-manufacturer causes a need for high throughput rates. In order to cope with such an environment, it is nowadays well recognized that the availability of automated planning procedures is a major asset.

In this paper we describe a typical case in PCB-assembly arising at a plant of Philips NV, a major PCB-manufacturer. We propose a planning procedure for a situation where a family of different board types are to be produced by a line of different placement machines. This procedure is based on a hierarchical decomposition of the planning problem. Not surprisingly, most subproblems in this decomposition are already very hard in terms of computational complexity. Moreover, the size of the problems we consider prohibits the effective use of exact solution methods. In order to obtain good solutions in a reasonable amount of time, we solve the subproblems approximately using heuristics and local search methods. The resulting procedure is tested on real-life instances (made available to us by Philips), for which we are able to close about 70% of the gap between the previously best-known makespan and a (fairly) weak lower bound, thus reducing the current overall makespan by approximately 17%.

Let us now give a short description of how the assembly of PCBs is organized at the Philips plant under study. There are a number of assembly-lines, each consisting of a number of placement machines which place electronic components on bare boards. There are different types of boards and different types of components to consider. Each board type is assigned to a line of placement machines, that is, all boards of a type 'flow' through the machines of a specific line. The electronic components must be mounted at prescribed locations on a board. The set of locations to be served on a board as well as the type of components to be placed at each location depends upon the board type. In other words, for each board type, a set of locations, and for each of these locations the type of the component to be placed there, is given. Components of each type are delivered to the machine by means of a so-called feeder. Each placement machine is equipped with a feeder rack which holds the feeders. Further, the machine has some device - dependent upon the technology (see Section 2) - which is able to pick components from the feeders and place these components onto the board.

A production plan associated with this description should at least specify the following:

(1) a *partition* of the set of board types into *families* which are to be assigned to different lines of placement machines, and a *sequence* of the board types within each family, indicating in which order these board types will be produced,

(2) for each board type, a *partition* of the set of component locations on this board, that is a decision concerning which locations are going to be served by which machine,

(3) for each machine, a *feeder rack assignment*, that is an assignment of feeders to positions in the feeder rack,

(4) for each pair consisting of a machine and a board type, a *component placement sequence*, that is an order in which components are placed at the locations on this board that are served by this machine, and

(5) for each pair consisting of a machine and a board type, a *component retrieval plan*, that is, for each component on the board, a rule indicating from which feeder this component should be retrieved.

In this paper, we focus on problems (2)-(5), thus we deal with planning problems that arise when a given family of board types is assembled by a single line of placement machines. In Section 2 we give a precise description of these problems (including some of the technological features of the placement machines used), and in Section 3 we describe our solution procedure. Section 4 is devoted to the performance of our procedure on real-life instances and Section 5 contains the conclusions. The remainder of the current section is devoted to literature related to the assembly of PCBs.

In case one strives to minimize the makespan for a single machine producing a single board type, the planning problem above reduces to problems (3), (4) and (5) (feeder rack assignment, component placement sequence and component retrieval plan). A number of studies focus on problems (3) and (4) only, since problem (5) vanishes when precisely one feeder is available per component type. In such a case, the interaction between problems (3) and (4) is of crucial importance. This issue has been identified by Drezner and Nof [16] (see also Walas and Askin [26] for a similar application in the production of metal parts). An approach based on location theory is reported by Foulds and Hamacher [17]. Leipälä and Nevalainen [20] suggest a solution procedure based on heuristically solving a TSP and a quadratic assignment problem in an iterative fashion. Other heuristic approaches are described in Francis, Hamacher, Lee and Yeralan [18] (see also Viczián [24]) and Younis and Cavalier [27]. Ball and Magazine [8] show that, when a feeder rack assignment is given, an optimal component placement sequence can, in some situations, be found in polynomial time. Bard, Clayton and Feo [10] propose a planning procedure tailored for a specific placement machine, in which problem (5) is explicitly addressed. (see also Crama, Flippo, van de Klundert and Spieksma [13]). Further, Ahmadi, Grotzinger and Johnson [2] present a model which determines, among other parameters, with how many feeders of each type the placement machine should be equipped.

When the planning problem is extended to a line of placement machines producing a single board type, problem (2) comes up. Crama, Kolen, Oerlemans and Spieksma [14] and van Laarhoven and Zijm [19] each propose a solution procedure for a line of CSM-60 placement machines. Lofgren, McGinnis and Tovey [21] treat a case where a board is allowed to visit a machine more than once.

Relatively few published studies deal with the case where multiple boards types are to be assembled by one or more machines: we mention Carmon, Maimon and Dar-el [11], Balakrishnan and Vanderbeck [7] and Askin, Dror and Vakharia [6]. In this situation the following issues may appear (cf. problem (1)). Since the set of component types needed to produce all board types can be larger than the available capacity in the feeder racks, one may be forced to partition the set of board types into subfamilies which can be produced with a fixed feeder assignment. Thus, one may have to solve problems (2)-(5) a number of times during the planning period to produce all board types. This problem is addressed in Tang and Denardo [22], [23] and Bard [9]. A number of authors have further investigated this *batch selection* problem (see Crama [12] for further references). Alternatively, given a number of assembly-lines, one may try to solve problem (1) in such a way that each family can be produced on a line (which describes the situation at hand, see Section 2). Also, one may partly circumvent the problem by prescribing some feeders (the 'common' feeders) to be permanently assigned to certain positions in the rack,

whereas other feeders (the 'exotic' ones) are loaded as needed, see Carmon, Maimon and Darel [11], Balakrishnan and Vanderbeck [7] and Agnetis, Askin and Sodhi [1] for a description of this idea. Askin, Dror and Vakharia [6] study the assembly of multiple board types by multiple machines in an open shop (rather than flowshop) environment. Another issue which becomes apparent is that, when dealing with more than one board type, problem (3) becomes much more complicated. Indeed, most of the planning procedures described in the literature we mentioned here, attempt to determine a feeder rack assignment for which an optimal component placement sequence can be found. However, when dealing with multiple board types, one has to find a single feeder rack assignment such that a good placement sequence can be found *for each* board type (see sections 2 and 3).

Finally, the technology employed by the placement machine under consideration may give rise to specific planning problems (see for instance Ahmadi and Kouvelis [4]). An overview of issues which arise in the operational planning of PCB assembly can be found in Ahmadi [3], Crama, Oerlemans and Spieksma [15] and Voogt [25].

2 Problem description

In this section we focus on a precise description of problems (2)-(5) for the situation encountered at a plant of Philips. Subsection 2.1 refines the description of the setting given in the introduction. Subsection 2.2 is devoted to the technological features of the placement machines under consideration.

2.1 Properties of the assembly environment

This subsection deals with the following issues:

- we motivate the choice of our objective function,
- we discuss the nature of the feeder rack assignment problem for our situation, and
- we describe the so-called component retrieval problem.

At the plant investigated, several lines of placement machines are devoted to the assembly of PCBs. Production is mostly organized in such a way that changing feeders for other purposes than refilling should not occur. Thus, the capacity of the feeder racks in the assembly-line restricts the set of board types which the line can handle: this available capacity should be large enough to accommodate all feeders needed to assemble the family of board types assigned to that line. Accordingly, at these plants, problem (1) mentioned in Section 1 is reformulated to take this restriction into account. From now on we assume that problem (1) has been solved; thus, we deal with a set of different types of PCBs (a family) that has to be produced by a line of several placement machines (sometimes referred to as the *flowshop*) without any feeder changes.

It is customary at the plant investigated to produce in *batch* mode. This means that, within a family, all boards (several hundreds) of one type are assembled consecutively before switching to another board type. Here again, we assume that the sequence of batch types is exogeneously determined (that is, in Step (1)). Due to competition and efficiency incentives, it is important to achieve high throughput rates for the batches. Obviously, the throughput rate of each batch

is determined by a machine in the line on which the makespan of this board is maximal, called the *bottleneck* machine. Therefore we choose as objective to minimize the sum over all board types of the makespans of these board types on their bottleneck machines (in the software used by Philips this objective function is only implicitly used). Of course, different types of PCBs, and therefore different batches, may have different bottleneck machines. Thus, more formally, let $t_m^p(s)$ denote the makespan of a board of type p on machine m for some given solution (i.e. production plan) s. With S denoting the set of feasible solutions, our objective function may be specified as:

$$\min_{s \in S} \sum_p \max_m t_m^p(s).$$

This seems a reasonable objective function when the batches are of approximately equal size. Otherwise, weights reflecting the size of the batches can be incorporated to obtain a more appropriate objective function.

Consider now the feeder rack assignment problem. As mentioned in the introduction, the feeder rack assignment problem becomes harder when multiple board types are involved. In fact, as far as we know, the feeder rack assignment problem with multiple board types has not been addressed explicitly in the literature. Of course, a straightforward way to deal with this problem is to reduce the multiple board case to the single board case. This could be done by creating a so-called *composite* board type which would consist of all the individual board types superposed on each other. In other words, a fictitious board type is made on which all locations of all board types of the family occur. Next, one can apply any existing software for solving the traditional feeder rack assignment problem for this composite board. In fact, this strategy is currently used in practice. Our approach takes a different point of view. A main characteristic of our solution procedure is to take into account as much as possible the *individual* board characteristics. This approach can be motivated by observing that although the set of component types needed for different board types in the family may be similar, the distribution of the locations to be served on those board types can be quite different. Our solution procedure, which is also based on machine characteristics to be discussed later, is described in Section 3.

Finally, consider the following issue. Imagine, for reasons of simplicity, the problem of minimizing the makespan of a single board on a single machine. Obviously, a solution to the resulting planning problem must consist of a component placement sequence and a feeder rack assignment. However, when components of a same type are assigned to more than one feederslot, solving these two subproblems is not sufficient. In addition, we have to decide for each component *from which feeder* it is to be retrieved. Of course, different decisions for a specific component may result in different makespans for the board. This issue is raised in Bard, Clayton and Feo [10], and the resulting *component retrieval problem* is further investigated in Crama, Flippo, van de Klundert and Spieksma [13]. A solution to this problem consists in a mapping from the set of board locations to the feeders (where the image of each location is simply the feeder delivering the component to be placed at this location). Call such a mapping a *component retrieval plan*. Then, the component retrieval problem can be stated as follows:

Given a component placement sequence and a feeder rack assignment, what component retrieval plan minimizes the makespan of the PCB?

Now, the way in which we are able to solve the component retrieval problem (and, in fact, some other issues) depends to some extent on the technological features of the placement machines used. We describe these machines in more detail in the next subsection.

2.2 The placement machine

In this subsection we describe the Fuji CP-IV placement machine that is used at the plant investigated. This description will enable us to translate the component retrieval problem for a Fuji CP-IV into a graph-theoretical problem (see Subsection 3.2).

Obviously, the task of any PCB assembly machine is to *place* components on a PCB. These components are to be retrieved or *gripped* from feeders. Apart from differences in techniques for gripping and placing (insertion, onsertion, glueing), placement machines differ in the way the coordination between placing and gripping activities is achieved. At the plants we consider, two types of placement (onsertion) machines are used, Fuji CP-III's and Fuji CP-IV's. For our purpose, it is sufficient to assume here that the Fuji CP-III operates in an identical fashion as the Fuji CP-IV, but at a different speed.

To perform its task, the CP-IV is equipped with a gripper, a placer and a carousel. The gripping of components from a feeder is performed by the gripper, and the placement is performed by the placer. The gripper as well as the placer do not move. Each time a component is gripped (or placed), the feeder rack (or the table containing the PCB) is positioned accordingly beneath the gripper (placer). The coordination of the interaction between the gripper and placer is achieved through the carousel or CAM. The carousel has 12 positions configured in a circle and it rotates clockwise in small rotations such that after 12 rotations it has rotated 360 degrees. See Figure 1 for a schematic representation of the CP-IV.

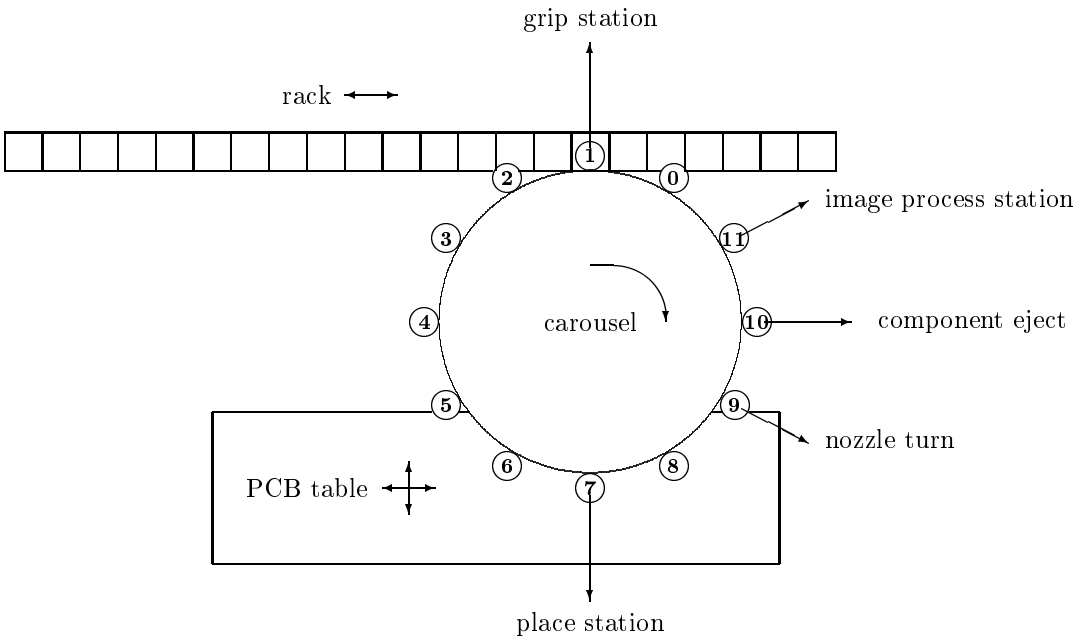

Figure 1: The Fuji CP II.

Let us now describe how the machine operates. Between two consecutive rotations a compo-

nent is gripped and a component is placed simultaneously. So, after say $i-1$ rotations ($i \geq 7$), two things happen at the same time: the i-th component to be gripped is gripped and stored in position i mod 12 of the carousel, and the $i-6$-th component to be placed is placed in the PCB from position $(i-6)$ mod 12 of the carousel. Observe that the gripper is six components ahead of the placer. Since the gripping of component i and the placement of component $i-6$ start (and end) at the same time, this implies that, as soon as the grip and the place activities have been performed, three devices start to move at the same time: the table moves to bring the next location beneath the placer, the rack moves to bring the next feeder beneath the gripper and the carousel rotates. The movement which takes maximum time determines the moment when a new iteration starts.

Notice that the modus operandi of the Fuji CP-IV described here (which was communicated to us by Philips) differs slightly from the one given in Bard, Clayton and Feo [10]. In our situation, the start of a grip activity coincides with the start of a place activity (at least after the first 6 rotations). This requirement is not present in the description given in [10]. It turns out that this requirement allows us to model the component retrieval problem straightforwardly as a shortest path problem (see Subsection 3.2), whereas in the absence of this requirement this is no longer the case (cf. Crama, Flippo, van de Klundert and Spieksma [13]).

As a final remark to this section, the reader will understand that the description above, although perhaps detailed, is not an exact replica of the truth. For instance, due to the fact that the speed of the carousel depends on the type of the components it carries, not all carousel rotations last equally long. Also, components which have to be rotated may take some extra time of the placer. Further, there are differences between a Fuji CP-III and a Fuji CP-IV placement machine besides speed. Although it is possible to model the aforementioned and other physical characteristics, we have chosen not to do so, for reasons of simplicity and since most of these characteristics will have only marginal effects on the makespans.

3 The planning procedure

Let us now return to the general planning problem. Summarizing the discussion in the previous sections, we want to find:

1. for each machine, a feeder rack assignment,
2. for each board type, a component placement sequence on each machine, such that for each PCB of that type, the sequences form a partitioning of the components required by the PCB,
3. for each pair consisting of a machine and a board type, a component retrieval plan.

In this section we describe our planning procedure. This procedure is divided into two phases: Phase 1 determines a feeder rack assignment for each machine, and Phase 2 produces, for each pair consisting of a machine and a board type, a component placement sequence and a component retrieval plan, given the feeder rack assignment of Phase 1. Accordingly, this section contains two subsections each devoted to the description of a phase in the planning procedure.

Clearly, the planning procedure we present is hierarchical in nature: first, a feeder rack assignment is computed, next a component placement sequence and component retrieval plan are determined. Of course, the following question then arises: how do we evaluate feeder rack

assignments computed during Phase 1 without computing a placement sequence and retrieval plan, that is, without solving Phase 2? We deal with this issue by computing an *estimate* of the makespan of each board type on each machine given the feeder rack assignment and a corresponding partition of the components of each board type (see (2) in Section 1). These estimates are then used as an indication of the quality of the feeder rack assignments found during the execution of the algorithm.

Before describing Phase 1 in more detail, let us first start with some observations related to the placement machines described in the previous section.

Observation 1: A feeder is nothing but a tape containing components of a certain type. These tapes are expensive, and to keep inventory of these tapes low, it is desirable to restrict the number of tapes containing components of a certain type. In the plant considered, often a bound of 2 feeders for each component type was imposed, that is, no more than 2 feeders with the same component type can be used on the line. We adopt this restriction in our planning procedure.

Observation 2: A basic characteristic, which has been observed by other authors as well (see e.g. Ahmadi, Grotzinger and Johnson [2] and Bard, Clayton and Feo [10]) concerns the so-called *free movement*. To explain this, consider again Figure 1. Between two consecutive grip (or place) activities, the carousel must rotate, and this takes a certain amount of time. During this time, the feeder rack, as well as the table containing the PCB, can move without increasing the makespan. Since it is impossible to avoid carousel rotations, this phenomenon occurs between each pair of consecutive grip (or place) activities. These movements of the feeder rack and the table, taking place during a carousel rotation, are referred to as free movement. Of course, the significance of this effect depends on the magnitude of the free movements. However, it can be considerable. In our situation, free movements of the feeder rack correspond to 1 position on the rack, that is, repositioning the feeder rack by 0 or 1 feeders is free. Concerning the table, free movement corresponds to approximately 2 cm on the table containing the PCB. Since the average PCB is approximately 20×30 cm, and the feeder rack contains mostly about 100 feeders, long table movements are less time consuming than long feeder rack movements. Hence we restrict the planning procedure to considering solutions in which all feeder rack movements are short, expecting that given this short feeder rack movements we can find a component placement sequence in which the table movements are not too long either. More specifically, we assume in Phase 1 that the predecessor of a component retrieved from some feeder i, is retrieved from feeder $i-1$, feeder i itself, or feeder $i+1$ (thereby utilizing the free movement as much as possible as far as the feeder rack is concerned). The solutions we obtain show that it is indeed possible to construct feeder rack assignments and component placement sequences such that subsequent components are mostly retrieved from consecutive feeders while the larger part of the table movements is free.

3.1 Constructing a feeder rack assignment: Phase 1

Let us now describe Phase 1. It consists of five steps.

Step 1: Determine which component types will have 2 feeders in the flowshop (see Observation 1).

Step 2: Decide, for each feeder, which locations it serves on each board type.

Step 3: Construct an arbitrary feeder rack assignment.

Step 4: Estimate the makespan for each board type on each machine, given the current feeder rack assignment.

Step 5: If some stopping criterion is satisfied, exit. Else, improve the feeder rack assignment using local search and go to Step 4.

Consider Step 1. In view of Observation 2 above, it is desirable for components that are within free gripping movement of each other, to be not too far apart from each other on the board, since otherwise the board movement (between consecutive placing operations) may take a long time. Thus, if there are more magazine rack positions than component types, one way to avoid long table movements is to assign two feeders to component types whose components are, on some boards, far apart. Of all strategies we have tested to utilize this idea the following simple strategy performed best. Compute, for each combination of board type and component type, a short Hamiltonian path through the corresponding locations using some (TSP) heuristic (we use farthest insertion). For each component type t, let l_t be the length of the longest edge occurring in some path corresponding to component type t. Next, we list the component types in order of decreasing l_t value and we assign two feeders to as many component types as possible, starting at the top of the list and proceeding downward, until total capacity of the feeder racks is exhausted (or until each type has two feeders). In this way it is decided which component types have more than one feeder. Notice that we use individual board type characteristics to select those component types.

Step 2 resembles the component retrieval problem. The difference is that the position of the feeders in the rack is not fixed yet. To partition the locations corresponding to components of each type for which there are two feeders placed in the rack, we propose the following. First, we consider a board type which has led us to assign two feeders to a component type t: consider board type p on which the Hamiltonian path as computed in Step 1 contains an edge of length l_t. Removing this longest edge partitions the locations corresponding to components of type t on board type p into two subsets, say $L_1^{t,p}$ and $L_2^{t,p}$.

We have to decide next how to partition the components of this type on boards other than boards of type p. We take the following approach. For each location corresponding to a component of type t not on board type p, we determine the minimal distance to some location in $L_1^{t,p}$ and, similarly, we determine the minimal distance to some location in $L_2^{t,p}$. Next, we assign this location to the subset whose corresponding minimal distance is minimal. (Notice that we assume here that a distance is known between two locations which do not occur on the same board type. These distances are computed as if the components where on a same board, as e.g. the composite board mentioned earlier. This composite board can be constructed unambigously since all component locations are expressed in terms of coordinates, and the position of the boards on the table are given.) This approach has outperformed several alternative approaches in our computational experiments. Its success should be contributed to the fact that in this way, for each board type, the set of locations to be served by a feeder lies in the same part of the board.

In this way, we obtain two sets of locations for each component type that has two feeders in the flowshop. More precisely, we refer to this partitioning of locations corresponding to components of a certain type as a *clustering*, and we refer, informally, to all components that are to

be retrieved from the same feeder as a *cluster*. In Figure 2 we give a more formal description of the algorithm described in steps 1 and 2.

In Step 3 we simply assign each feeder arbitrarily to some position in the racks of the machines such that a feasible feeder rack assignment is obtained.

Consider Step 4. Recall from Observation 2 that, given a feeder rack assignment, we intend to construct component placement sequences with the property that the rack moves at most one position between any two consecutive grip activities. In addition, when estimating the makespan of a board given a feeder rack assignment, we assume that one pass of the feeder rack through all feeders gives a good approximation of an optimal way of moving the feeder rack. More explicitly, we estimate the makespan by assuming that the feeder rack movement in an optimal solution follows approximately the following pattern: the rack starts at the feeder in the left most position, to be called feeder 1. Then, all components are gripped (and placed) of the cluster assigned to feeder 1, interleaving them by gripping (and placing) components of feeder 2, ending with a grip of a component from feeder 2. This is followed by gripping (and placing) the remainder of the components that are to be gripped from feeder 2, interleaved with the gripping of components of feeder 3 et cetera.

This leads to a solution in which all feeder rack movements are free (if at least one component is retrieved from each feeder). Therefore, the makespan depends on the length of the table movements only, i.e. the length of a Hamiltonian path through the locations, that respects the pattern of the feeder rack movements described above. For our purposes it is desirable to be able to compute quickly an estimate of the length of such a Hamiltonian path. Therefore we propose the following method, that does not use the exact feeder rack assignment, but only knowledge of which feeders are assigned to adjacent positions in the feeder rack. Notice that for every pair of feeders there is a set of locations on a board where the components from these feeders will be placed. We compute, for every pair of feeders, a value which equals the length of a Hamiltonian path through these locations, thereby utilizing the fact that they may be gripped interleavingly. Notice that these quantities can be computed independently of a feeder rack assignment, reducing the amount of computations required to evaluate individual feeder rack assignments.

Of course, it may well be the case that, given a feeder rack assignment and some board type, there is a set of adjacent feeders from which no component is taken at all. In that case, the feeder rack movement is not free, and the makespan depends on the duration of this feeder rack movement. We next present an algorithm to estimate the makespan of every board type in the family for a given feeder rack assignment on some machine.

We compute, for each board type p in the family, two so-called cluster distance matrices D^p and E^p. D^p_{ij} represents the length of a (short) Hamiltonian path through all locations of clusters C_i and C_j that are to be placed on p. E^p_{ij} is the maximum of two numbers, E^{p1}_{ij} and E^{p2}_{ij}, where E^{p1}_{ij} is the minimum distance over all distances between locations of C_i and C_j on board type p and E^{p2}_{ij} corresponds to the distance between the feeders positioned in the rack which correspond to clusters C_i and C_j. Notice that these computations must be performed efficiently, since for the problem instances described in Section 4, the computation of the matrices D^p requires constructing several hundreds of thousands Hamiltonian paths.

Recall that the order in which the feeders are placed on the rack is known and let us refer

L^t: set of locations corresponding to components of type t.
$L^{t,p}$: set of locations corresponding to components of type t on a board of type p.
$P^{t,p}$: sequence of locations corresponding to components of type t on a board of type p induced by a Hamiltonian path.
$d_{ij}^{t,p}$: distance between location i and location j corresponding to components of type t on a board of type p.
$dis(i, L)$: minimum distance over the locations in the set L between location i and a location from L.
$\#t$: number of component types.
$\#p$: number of board types.
cap: number of feeder rack positions in the line.

1. for $t := 1$ to $\#t$ do
   for $p := 1$ to $\#p$ do
   construct a Hamiltonian path $P^{t,p}$ using distance matrix $d^{t,p}$;
   let the value of the longest edge in $P^{t,p}$ be denoted by $z_{t,p}$;
2. for $t := 1$ to $\#t$ do
   (a) $l_t :=\max_p z_{t,p}$;
   (b) $p_t :=\arg\max_p z_{t,p}$;
3. sort the component types in decreasing order of l_t and index them $1, 2, \ldots, \#t$;
4. $t^* := \min(\text{cap}, 2 \times \#t) - \#t$; *comment: t^* corresponds to the number of feeder duplications*
5. for $t := 1$ to t^* do
   (a) partition L^{t,p_t} into L_1^{t,p_t} and L_2^{t,p_t} by removing the longest edge from P^{t,p_t};
   (b) $C_{2t-1} := L_1^{t,p_t}$; $C_{2t} := L_2^{t,p_t}$;
   (c) for each $i \in \cup_{p \neq p_t} L^{t,p}$ do
   if $dis(i, L_1^{t,p_t}) < dis(i, L_2^{t,p_t})$ then $C_{2t-1} := C_{2t-1} \cup \{i\}$ else $C_{2t} := C_{2t} \cup \{i\}$;
6. for $t := t^* + 1$ to $\#t$ do $C_{t^*+t} := L^t$;

Figure 2: Algorithm for steps 1 and 2

to a feeder from which components are gripped in order to assemble a board of type p as an *active* feeder. Further, each feeder, except the leftmost and the rightmost ones, is adjacent to two other feeders to which we will refer as its *left neighbor* and its *right neighbor* respectively. Now, to estimate the makespan of a board of type p on a machine we do the following.

First, consider for each active feeder, its left neighbor. If its left neighbor is active, we choose from the D^p matrix the entry corresponding to these two feeders (the value of this entry equals the length of a Hamiltonian path through the locations of both corresponding clusters). If the left neighbor is not active, we choose the diagonal element of D^p corresponding to the current active feeder (the value of this entry equals the length of a Hamiltonian path through the locations of the corresponding (single) cluster). Second, consider for each active feeder its right neighbor. If it is active we do nothing, else we again choose the diagonal element of D^p corresponding to the current active feeder. Finally, we sum all chosen entries. Observe now that each location is part of two Hamiltonian paths. To correct for this we simply divide the computed sum by 2. In this way we estimate the time spent on table movements.

In order to estimate the time spent on rack movements we identify pairs of active feeders which are consecutive but not adjacent. For each such pair we find the corresponding entry in E^p, then we sum these values and use this sum to estimate the time spent on feeder rack movements. A more formal description of the algorithm estimating the makespan of board type p on a machine with a given feeder rack assignment is as given in Figure 3.

Using the algorithm in Figure 3, we get an estimate of the makespan of every board type for a given feeder rack assignment for each machine in the flowshop, without having to specify a component placement sequence. Based on these estimated makespans, we can also get an estimate of the objective function for a given feeder rack assignment for each machine. Of course, the accuracy of such an estimate depends on the quality of the Hamiltonian paths constructed, and on the component placement sequences that will eventually be found. (We will see later that the estimates are usually extremely good (see Table 2 in Section 4)).

Step 5 of our solution procedure optimizes the feeder rack assignments, using the estimated objective function value found in Step 4. In fact, it is in this step that our estimates are used extensively. Throughout this step objective function evaluations are based on the estimates of the makespans, instead of the actual makespans. We try to optimize the feeder rack assignment by using two heuristics alternately.

One heuristic tries to exchange between two machines a pair of clusters, together with the corresponding feeders, to better balance the workload. To determine which pair of machines are candidates for exchanging clusters we do the following. For each pair of machines we sum over all board types the absolute difference of the respective processing times. Next, we select that pair of machines for which this sum is maximal, and attempt to improve our current feeder rack assignment by exchanging feeders (and corresponding clusters) between these machines. Now what do we mean by improve? Obviously, what we want to improve is the objective function as described in Subsection 2.1. However we have chosen another surrogate objective function to speed up the heuristic. Let M_i and M_j be the machines between which feeders are being exchanged, then the surrogate objective function takes on the value which the real objective function would have taken, had machines M_i and M_j been the only machines in the flowshop.

The other heuristic reoptimizes the feeder rack assignment for a single machine. More precisely, for a given machine, it attempts to minimize the sum of the makespans of the boards on that machine, by optimizing the feeder sequence. This sequencing problem is solved using an insertion heuristic in which the makespan estimation algorithm given in Figure 3 is used to

π: a permutation of the feeders, i.e the order in which the feeders are placed on the rack (say from left to right),
C_i^p: the set of components from cluster C_i (the cluster corresponding to the feeder positioned in $\pi(i)$) that have to be placed on board type p.

1. $i := 1, span := 0$;
2. while $|C^p_{\pi(i)}| = 0$ do $i := i + 1$;
3. $span := span + 1/2 \times D^p_{\pi(i),\pi(i)}$;
   $i := i + 1$;
4. while $|C^p_{\pi(i)}| > 0$ do
   (a) $span := span + 1/2 \times D^p_{\pi(i-1),\pi(i)}$;
   (b) $i := i + 1$;
   (c) if $i =$ total number of feeders then set $k = i$ and goto step 7;
5. $k := i - 1$;
   $span := span + 1/2 \times D^p_{\pi(k),\pi(k)}$;
6. while $|C^p_{\pi(i)}| = 0$ do
   (a) $i := i + 1$;
   (b) if $i =$ total number of feeders then goto 7;
7. $span := span + E^p_{\pi(k),\pi(i)}$;
   goto 3;
8. $span := span + 1/2 \times D^p_{\pi(k),\pi(k)}$;

Figure 3: Makespan estimation algorithm

evaluate the insertions.

Together these two heuristics deliver better solutions faster than other approaches we have tested. Since the heuristics work with different objective functions, which are in turn different from the original objective function, the process of calling both heuristics in turns does not neccesarily converge. Therefore we have the following stopping criterion. We specify some upperbound (say 5 secs.), and the algorithm stops if after some iteration, the maximum over all pairs of machines of the sum of the absolute differences of their makespans does not exceed this upperbound. This strategy implies that when the algorithm stops, there may still be some room for improvement of the balance. Achieving these last tenths of percents of improvement is time consuming, and becomes less attractive as one notices that this phase of the algorithm uses estimations rather than the actual makespans.

3.2 Constructing a component placement sequence and a component retrieval plan: Phase 2

Let us now describe Phase 2. It consists of three steps.

Step 1: Determine, for each machine - board type combination, a component placement sequence.

Step 2: Determine, for each machine - board type combination, a component retrieval plan.

Step 3: Improve the component placement sequence using local search. If no improvements are found, stop, else go to Step 2.

Consider Step 1. For a given board type and machine, we construct an initial component placement sequence as follows. First assume that each location is served from the feeder associated to the cluster containing that location. For the first two feeders, determine two locations, c_1 (from the left most feeder, feeder 1) and c_2 (from the feeder adjacent to it, feeder 2), that are nearest to each other. Then sequence before c_1, all locations that are to be served from the left most feeder as well, using some insertion heuristic. Next determine the two locations c_2' from feeder 2, and c_3' from feeder 3, that are nearest to each other, and sequence all remaining locations that must be served from feeder 2 between c_2 and c_2' et cetera. Given this sequence we solve the component retrieval problem (see below), which provides a first solution.

Consider now Step 2. How do we solve the component retrieval problem? In order to answer this question, recall that, when solving this problem, we assume that a component placement sequence and a feeder rack assignment are given. We are going to construct a graph G and show that the component retrieval problem is equivalent to finding a shortest path in this graph.

We assume that the components are placed in numerical order (so component i refers to the i-th component in the component placement sequence). Furthermore, we assume that following the gripping of the last component, six more components are to be gripped, from the same feeder from which the last component was gripped. These additional 'dummy' grips are performed in parallel with the last six places, and do not increase the makespan since the feeder rack need not be repositioned to perform them. Similarly we assume that there are six dummy place activities to accompany the first six grip activities, that also do not increase the makespan. So, between each two consecutive carousel rotations, both a grip and a place operation must take place. We

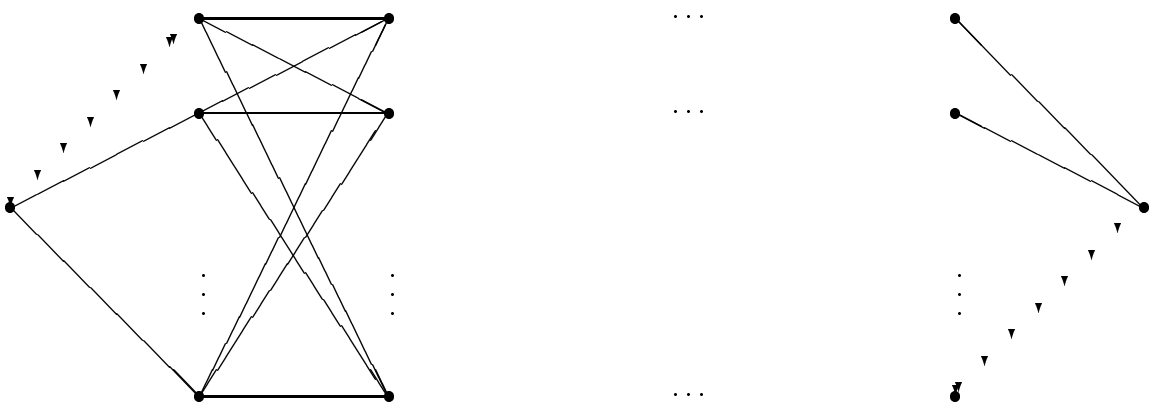

Figure 4: Graph G

define n to be the number of such pairs. (Notice that n exceeds the number of components by six.) Further, let K equal the number of feeder rack positions.

For ease of exposition, we refer to the location on the board corresponding to component i as location i, $1 \leq i \leq n$. Let

Δgp	= time needed to grip (and hence place) a component,
Δc	= time needed for a carousel rotation,
$\Delta t(i, i+1)$	= time needed for a table movement to bring location $i+1$ beneath the placer, starting from a position in which location i is beneath the placer, $1 \leq i \leq n-1$, and
$\Delta f(r, s)$	= time needed for a rack movement to bring feeder rack position r beneath the gripper, starting from a position in which feeder rack position s is beneath the gripper, $1 \leq r, s \leq K$.

We construct a graph G as follows. Let $F_i \subseteq \{1, \ldots, K\}$ be the index set corresponding to feeder rack positions which hold feeders from which component i may be retrieved. The graph has a source, a sink and n intermediate layers. Each layer i contains $|F_i|$ vertices, denoted by $v_j^i, j \in F_i, 1 \leq i \leq n$. Indeed, there is one vertex in layer i for each feeder from which component i may be retrieved. The interpretation of vertex v_j^i is the start of the grip (and place) activity which grips component i from feeder j. Each vertex in layer i has an arc going to each vertex in layer $i+1$, the source has an arc going to each vertex in layer 1 and each vertex in layer n has an arc going to the sink. There are no other arcs. See Figure 4 for a representation of G. The weight of an arc emanating from the source is 0, and the weight of an arc going to the sink is Δgp. The weight of an arc from v_r^i to v_s^{i+1}, $r \in F_i, s \in F_{i+1}, 1 \leq i \leq n-1$, equals:

$$\Delta gp + \max(\Delta c, \Delta t(i, i+1), \Delta f(r, s)).$$

Notice that when one interprets vertex v_r^i as the start of the grip (and place) activity which grips component i from feeder r, the weight of an arc defined above is equal to the time between two consecutive grips. (This follows from the description in Subsection 2.2).

Consider now any path in the graph G from the source to the sink. This path contains one vertex from each layer, reflecting a choice of feeders for the retrieval of the n components. Further, since the weight of an arc corresponds to the time between two consecutive grip activities, the length of the path equals the makespan of the assembly of the board. Also, it is easy to verify that each solution of the component retrieval problem corresponds to a unique path in G. Since one is interested in the shortest makespan, it follows that the component retrieval problem reduces to a shortest path problem on G, for which efficient algorithms are available (see for instance Ahuja, Magnanti and Orlin [5]). (The recursive formulation given in Bard, Clayton and Feo [10] would also lead immediately to a polynomial time algorithm for this version of the component retrieval problem.)

Finally, we restrict ourselves here to noticing that, even for a given component placement sequence and a feeder rack assignment, the computation of the makespan of a PCB is in general a nontrivial task. In fact, for other technologies the component retrieval problem may become substantially more difficult (see Crama, Flippo, van de Klundert and Spieksma [13]).

In Step 3, we try to improve the placement sequence by TSP-like local search techniques, using 2opt and a restricted version of 3opt. This local search process may be very time consuming since each local search step requires solving the component retrieval problem. Therefore we have sought ways to speed up this local search phase. An easy way of doing so, without substantially influencing the effectiveness of the 2opt heuristic, is to keep the component retrieval plan fixed. In this way, the time consuming resolving of the component retrieval problem in each iteration of the 2opt heuristic can be skipped. (It should be noted however, that each iteration still requires performing some non-negligible computations due to the fact that changes in the component placement sequence will change for some components, the component that is gripped while it is being placed.) On the other hand, especially for the restricted 3opt heuristic (that essentially takes out one component of the placement sequence and then tries to reinsert it) resolving the component retrieval problem may be well worth the additional effort. We have tried to reduce the running time of the local search heuristics by implementing several ideas that keep them from considering or evaluating (by solving the component retrieval problem) moves that will not result in an improvement.

We aimed to keep the running time of the entire algorithm (phases 1 and 2) within certain limits. As a consequence the algorithm cannot spend too much time optimizing the component placement sequence of each board, even though the local search heuristics usually improve the solutions significantly. The methods described above to speed up these heuristics ensure that the benefits of these heuristics are realized.

Notice that the feeder rack movements resulting from the final component placement sequences may not always utilize free movements as the intended feeder rack movements in Phase 1 of the solution approach did. Further, notice that the clustering found in steps 1 and 2 of Phase 1 can be changed by the component retrieval plan.

4 Computational results

In this section we test our planning procedure on two datasets. Dataset 1 corresponds to a family consisting of 9 board types assembled by a line of 3 CP-IV machines. Dataset 2 corresponds to a family consisting of 7 board types assembled by a line of 2 machines, a CP-IV and a

board type	Machine 1		Machine 2		Machine 3	
	NoC	time	NoC	time	NoC	time
1	50	13.8	52	13.5	49	13.5
2	49	13.8	52	13.5	50	13.8
3	86	21.1	86	21.2	84	21.2
4	66	17.4	72	18.4	70	18.5
5	25	7.9	28	8.0	25	7.0
6	25	7.9	28	8.2	26	7.5
7	42	11.3	43	12.2	47	13.4
8	58	15.7	61	16.1	60	16.9
9	304	69.4	293	69.3	332	71.0

	time
Total makespan	184.8
Current solution	244.1
Lower bound	154.8

Table 1: Final results Dataset 1

CP-III machine. These datasets are real-life data made available to us by Philips. The planning procedure we described above is programmed in Turbo Pascal and run on a personal computer with a 486 33MHz processor. The results of our procedure are described in Tables 1, 2 and 3.

To explain Table 1, consider a column corresponding with a machine. An entry in this column has two numbers: "NoC" is the number of components of the specific board type placed by that machine, and "time" equals the number of seconds it takes to place these components by this machine. The total makespan is computed by summing over the board types, the makespan of these boards on their bottleneck machines. This total makespan is compared with the makespan of the solutions obtained by Philips (referred to as current solution) and with a lower bound. This lower bound is computed as follows. Let Δc represent the time a single carousel rotation takes on the fastest machine in the line; further, let *totcomp* be the total number of components to be placed to produce a single board of each type in the family, and let m equal the number of machines in the line. Then $\frac{\Delta c \cdot totcomp}{m}$ is a valid lower bound for the makespan. For dataset 1, we are able to improve the current solution by almost a minute, closing 66% of the gap between the lower bound and the current solution.

In Table 2 the estimates of the makespans as computed in Phase 1 are presented. We conclude that these estimates are accurate enough (usually within a few percent) to give a realistic impression of the actual makespans delivered by Phase 2. This might for example be useful to evaluate alternative solutions to the partitioning of board types into families (problem (1) in Section 1).

Table 3 consists of four subtables, arising as follows. As mentioned in Subsection 2.2, the difference between a CP-III and a CP-IV is its speed. To model this difference, we assume that speed (CP-IV) = speedfactor * speed (CP-III). Since this speedfactor (spf) is a simplification

board type	Machine 1		Machine 2		Machine 3	
	actual	estimated	actual	estimated	actual	estimated
1	13.8	13.6	13.5	13.1	13.5	13.5
2	13.8	13.5	13.5	13.1	13.8	13.6
3	21.1	22.2	21.2	21.3	21.2	22.2
4	17.4	18.8	18.4	18.5	18.5	19.2
5	7.9	8.2	8.0	8.4	7.0	7.4
6	7.9	8.1	8.2	8.4	7.5	7.6
7	11.3	12.3	12.2	12.8	13.4	13.3
8	15.7	16.4	16.1	17.2	16.9	17.0
9	69.4	71.2	69.3	72.2	71.0	71.6

Table 2: Estimated Results Dataset 1

spf = 1.15	Machine 1		Machine 2	
board type	NoC	time	NoC	time
1	107	28.7	129	29.8
2	138	37.1	164	37.1
3	137	37.2	162	37.0
4	306	80.7	378	82.6
5	118	31.3	139	31.6
6	149	39.2	174	39.5
7	150	39.5	172	38.8

	time
Total makespan	297.3
Current solution	361.8
Lower bound	284.2

spf = 1.20	Machine 1		Machine 2	
board type	NoC	time	NoC	time
1	104	29.4	132	31.0
2	134	37.3	168	39.1
3	131	37.2	168	38.9
4	303	82.2	381	84.6
5	112	31.7	145	33.7
6	144	40.4	179	40.9
7	144	39.4	178	40.8

	time
Total makespan	309.0
Current solution	361.8
Lower bound	289.8

spf = 1.25	Machine 1		Machine 2	
board type	NoC	time	NoC	time
1	102	30.2	134	32.1
2	132	38.2	170	39.4
3	131	38.2	168	39.1
4	299	84.1	385	86.4
5	110	33.0	147	34.2
6	142	41.5	181	41.5
7	140	41.1	182	41.3

	time
Total makespan	314.0
Current solution	361.8
Lower bound	295.2

spf = 1.30	Machine 1		Machine 2	
board type	NoC	time	NoC	time
1	99	31.3	137	31.1
2	128	39.5	174	39.1
3	126	39.2	173	38.7
4	285	84.6	399	87.7
5	105	31.1	152	34.3
6	138	42.7	185	41.4
7	137	41.0	185	41.4

	time
Total makespan	315.9
Current solution	361.8
Lower bound	300.3

Table 3: Final results Dataset 2

of reality it is hard to estimate exactly. The speedfactor is believed to be approximately 15%, but Table 3 also shows the outcomes for other values. Although it is hard to make an exact statement regarding the outcomes of our algorithm based on Table 3, it shows that our planning procedure is quite robust. The makespan grows more or less proportionally to the speedfactor. Moreover, each of the four solutions exceeds the lower bound by not more than 7%, while the current solution exceeds the lower bound by more than 20%.

In our view, the main difference between the approach presented here and the approach used to obtain the solutions currently used in practice is that we try to use as much as possible the individual board type characteristics, as opposed to the existing software which uses a composite board (see Section 2). Hence, we believe that this difference in solution approach causes, at least for a large part, the gap between the solutions found by the two approaches. How can we test this hypothesis? If it were true, then it would imply that, for families consisting of a single board type, the solutions found by the two approaches should not differ as much as they do for larger families. Thus, to substantiate our claim that individual board type characteristics matter, we performed the following experiment (see Table 4). We chose board type no. 9 of dataset 1 and considered this board type once as a family (see the row in Table 4 denoted by "sole member") and once simply as part of its original family (see the row in Table 4 denoted by "part of family"). For both cases we applied our approach (see the column in Table 4 denoted by "individual approach") and the approach currently used at Philips (see the column in Table 4 denoted by "composite approach").

board type no. 9 of dataset 1	individual approach	composite approach
sole member	70.0	74.3
part of family	71.0	79.4

Table 4: Comparison between the two approaches for board no. 9 of dataset 1

The results in Table 4 seem to support our claim. Indeed, in our approach, the makespan barely increases (1 second) when the board is handled as a member of a large family rather than by itself. In the composite board approach, this increase amounts to 5 seconds. Alternatively viewed, the individual approach gives a makespan for board no. 9 as part of its family which is better by 8.4 seconds than the solution found for this board by the composite approach. Only 4.3 seconds (the difference between the makespans found for board 9 when considered as a family) of this improvement can be attributed to better search techniques, and thus almost 50% of the total improvement is due to the difference in the two approaches, or more concretely, to taking into account the individual board type characteristics.

A similar experiment was conducted with a board type of dataset 2. As in dataset 1, we observed that approximately 50% of the total improvement could be attributed to the individual board type characteristics. Thus, the results of these experiments tend to support the hypothesis.

Concerning the topic of running times, we restrict ourselves to the following general remarks. The running time of Phase 1 of the planning procedure (that is, constructing a feeder rack assignment) varies. Due to the fact that it is much harder to balance three machines than two, step 5 takes much more time for dataset 1 than for dataset 2. The exact running time

depends on the dataset and the stopping criterion, but Phase 1 takes approximately 15 minutes on a 33Mhz 486. For both instances, Phase 2 took roughly about 10 to 15 minutes. In fact, the running times given above are in the same order of magnitude as the running times of the existing software.

Summarizing, Tables 1, 2 and 3 show that the solution procedure we present yields, at least in terms of the makespan, significantly better results than the existing software. For a large part, this difference is caused by the fact that we solve the feeder rack assignment problem using individual board characteristics contrary to existing software which uses a composite board type (see Tables 4 and 5). Another (small) advantage of the solutions found by our approach is that the movements of the feeder rack tend to be relatively small. This causes less wear for the rack.

5 Conclusions

This paper deals with the assembly of a family of board types by a single line of placement machines. By decomposing the planning problem, a number of subproblems arise. An important subproblem is to construct a feeder rack assignment for each of the machines that allows us to construct good placement sequences for each of the board types in the family. Here, we explicitly address this problem and we propose a heuristic based on the individual board characteristics. This heuristic is incorporated into a solution procedure which delivers a solution for the general planning problem. Since we strived for running times of the same order as the existing software, we only considered very simple local search methods for some of the subproblems. The computational results show that this approach works well.

Acknowledgements:
We would like to thank Mr. Voogt and Mr. Driessen of Philips NV for their willingness to provide us with data and explanations and for their insightful comments on an earlier version of this paper.

The first author gratefully acknowledges the partial support of ONR (Grants N00014-92-J-1375 and N00014-92-J-4083) and NATO (CRG931531).

References

[1] Agnetis, A., R.G. Askin and M.S. Sodhi, Tool addition strategies for flexible manufacturing systems, *the International Journal of Flexible Manufacturing Systems* **6**, 1994, 287–310.

[2] Ahmadi, J., S. Grotzinger and D. Johnson, Component allocation and partitioning for a dual delivery placement machine, *Operations Research* **36**, 1988, 176–191.

[3] Ahmadi, R.H., A hierarchical approach to design, planning, and control problems in electronic circuit card manufacturing, in: *Perspectives in Operations Management*, R.K. Sarin, editor, Kluwer Academic Publishers, Dordrecht, 1993, 409–429.

[4] Ahmadi, R.H. and P. Kouvelis, Staging problem of a dual delivery pick-and-place machine in printed circuit card assembly, *Operations Research* **42**, 1994, 81–91.

[5] Ahuja, R.K., T.L. Magnanti and J.B. Orlin, *Network flows*, 1993, Prentice-Hall, Englewood Cliffs, New Jersey, USA.

[6] Askin, R.G., 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**, 1994, 587–608.

[7] Balakrishnan, A. and F. Vanderbeck, A tactical planning model for mixed-model electronics assembly operations, CORE Discussion paper 9349, Catholic University of Louvain, 1993.

[8] Ball, M.O. and M.J. Magazine, Sequencing of insertions in printed circuit board assembly, *Operations Research* **36**, 1988, 192–201.

[9] Bard, J.F., A heuristic for minimizing the number of tool switches on a flexible machine, *IIE Transactions* **20**, 1988, 382–391.

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

[11] Carmon, T.F., O.Z. Maimon and E.M. Dar-el, Group set-up for printed circuit board assembly, *International Journal of Production Research* **27**, 1989, 1795–1810.

[12] Crama, Y., Combinatorial optimization models for production scheduling in automated manufacturing systems, in *Semi-Plenary Papers of the 14th European Conference on Operational Research*, R. Slowinski, editor, Poland, 1995, 237–259; to appear in the European Journal of Operational Research.

[13] Crama, Y., O.E. Flippo, J.J. van de Klundert and F.C.R. Spieksma, The component retrieval problem in printed circuit board assembly, Research Memorandum RM/95/033, Faculty of Economics and Business Administration, University of Limburg, 1995.

[14] Crama, Y., A.W.J. Kolen, A.G. Oerlemans and F.C.R. Spieksma, Throughput rate optimization in the automated assembly of printed circuit boards, *Annals of Operations Research* **26**, 1990, 455–480.

[15] Crama, Y., A.G. Oerlemans and F.C.R. Spieksma, *Production planning in automated manufacturing*, Lecture Notes in Economics and Mathematical Systems **414**, 1994, Springer-Verlag, Berlin, Germany.

[16] Drezner, Z. and S. Nof, On optimizing bin picking and insertion plans for assembly robots, *IIE Transactions* **16**, 1984, 262–270.

[17] Foulds, L.R. and H.W. Hamacher, Optimal bin location and sequencing in printed circuit board assembly, *European Journal of Operational Research* **66**, 1993, 279–290.

[18] Francis, R.L., H.W. Hamacher, C.-Y. Lee and S. Yeralan, Finding placement sequences and bin locations for cartesian robots, *IIE Transactions* **26**, 1994, 47–59.

[19] Laarhoven, P.J.M. van, and W.H.M. Zijm, Production preparation and numerical control in PCB assembly, *the International Journal of Flexible Manufacturing Systems* **5**, 1993, 187–207.

[20] Leipälä, T. and O. Nevalainen, Optimization of the movements of a component placement machine, *European Journal of Operational Research* **38**, 1989, 167–177.

[21] Lofgren, C.B., L.F. McGinnis and C.A. Tovey, Routing printed circuit cards through an assembly cell, *Operations Research* **39**, 1991, 992–1004.

[22] Tang, C.S. and E.V. Denardo, Models arising from a flexible manufacturing machine, part I: minimization of the number of tool switches, *Operations Research* **36**, 1988, 767–777.

[23] Tang, C.S. and E.V. Denardo, Models arising from a flexible manufacturing machine, part II: minimization of the number of switching instants, *Operations Research* **36**, 1988, 778–784.

[24] Viczián, I., Finding placement sequences and bin locations for cartesian robots, A working paper of the University of Würzburg, 1993.

[25] Voogt, S., Short term scheduling in PCB assembly, Philips Report CTR 597-93-0106, 1993.

[26] Walas, R.A. and R.G. Askin, An algorithm for NC turret punch press tool location and hit sequencing, *IIE Transactions* **16**, 1984, 280–287.

[27] Younis, T.A. and T.M. Cavalier, On locating part bins in a constrained layout area for an automated assembly proces, *Computers and Industrial Engineering* **18**, 1990, 111–118.