

The Component Retrieval Problem in Printed Circuit Board Assembly

YVES CRAMA

Ecole d'Administration des Affaires, Université de Liège, 4000 Liège, Belgium

OLAF E. FLIPPO

Department of Quantitative Economics, Faculty of Economics, University of Limburg, 6200 MD Maastricht, The Netherlands

JORIS VAN DE KLUNDERT

Department of Quantitative Economics, Faculty of Economics, University of Limburg, 6200 MD Maastricht, The Netherlands

FRITS C. R. SPIEKSMAS

Department of Mathematics, University of Limburg, 6200 MD Maastricht, The Netherlands

Abstract. Minimization of the makespan of a printed circuit board assembly process is a complex problem. Decisions involved in this problem concern the specification of the order in which components are to be placed on the board and the assignment of component types to the feeder slots of the placement machine. If some component types are assigned to multiple feeder slots, an additional problem emerges: for each placement on the board, one must select the feeder slot from which the required component is to be retrieved. In this paper, we consider this component retrieval problem for placement machines of the Fuji CP type. We explain why simple forward dynamic programming schemes cannot provide a solution to this problem, invalidating the correctness of an algorithm proposed by Bard, Clayton, and Feo (1994). We then present a polynomial algorithm that solves the problem to optimality.

The analysis of the component retrieval problem is facilitated by its reformulation as a PERT/CPM problem with design aspects: finding the minimal makespan of the assembly process amounts to identifying a design for which the longest path in the induced PERT/CPM network is shortest. The complexity of this network problem is analyzed, and we prove that the polynomial solvability of the component retrieval problem is caused by the specific structure it inflicts on the arc lengths of the network: in the absence of this structure, the network problem is shown to be NP-hard.

Key Words: PCB assembly, feeder assignment, dynamic programming

1. Introduction

The problem of determining optimal production plans for the automated assembly of printed circuit boards (PCBs) has been investigated by numerous researchers (see, e.g., Ahmadi 1993, and Crama, Oerlemans, and Spieksma 1996). A precise definition of this problem is highly dependent on the specific features of the assembly machines and, more generally, of the technological environment. As a rule, however, the problem is very complex, and for this reason, many authors have proposed to solve it by decomposing it into subproblems (see, e.g., Ahmadi 1993, Ball and Magazine 1988, Bard et al. 1994, Crama et al. forthcoming,

Crama et al. 1996, and van Laarhoven and Zijm 1993). Here again, the subproblems emerging from the decomposition vary according to the context, as do their computational complexity. In this paper, we concentrate on one such subproblem; namely, the component retrieval problem (CRP) that arises when the placement machine operates like a machine of the Fuji CP family. Briefly stated, CRP is defined to be the following problem: for a given placement sequence of components on the board and for a given assignment of component types to (possibly multiple) feeder slots of the placement machine, decide from which feeder slot each component should be retrieved.

In the next section, we describe the assembly process associated with a Fuji CP placement machine and the role of the component retrieval problem in this process. In section 3, we present a formulation of CRP in terms of a PERT/CPM network problem with design aspects; finding the minimal makespan of the assembly process amounts to identifying a design for which the longest path in the induced network is shortest. Alternatively, the component retrieval problem also may be viewed as a shortest path problem with (path-induced) side constraints.

In section 4, an example is discussed that reveals that a simple forward dynamic programming approach does not necessarily yield optimal solutions for the CRP. In fact, the example suggests that, to retain enough information to compute the optimal solution of the CRP, any dynamic programming procedure should involve a state-space whose size grows more than linearly with n , that is, the number of components to be placed. These negative observations, which invalidate in particular the forward dynamic programming approach proposed by Bard et al. (1994) may serve as a justification for the relatively complex solution algorithm presented in section 5. This algorithm is based on the network formulation of section 3 and can be viewed as a "two-phase" dynamic programming approach with *pairs* of grip activities defining the state-space. Therefore, the size of this state-space is *quadratic* in n .

Since the network optimization problems described in section 3 are of interest beyond the special case of the CRP, their time complexity is further investigated in section 6. We prove that the polynomial solvability of the CRP is due to the specific structure of the arc lengths in the PERT/CPM model of CRP; in the absence of this structure, the network problem is shown to be NP-hard. The paper concludes with a brief summary.

2. The placement machine

Assembling a printed circuit board consists in placing a number of electronic components, each of a prespecified type, at prespecified locations on a bare board. The placement machine considered in this paper is a member of the Fuji CP family (CP II, CP III, CP IV, . . .), yet our analysis may apply to other machines having similar characteristics, such as the Panasonic Mk1 considered by Horak and Francis (1995).

We now briefly describe the operating mode of the Fuji CP placement machine (we refer to Bard et al. 1994 and Crama et al. forthcoming for a more complete description of this process and related references). The machine is equipped with a magazine rack that contains a number of slots to which feeder tapes can be assigned. Each tape bears components of a single type, and feeder tapes with the same component type may be assigned to multiple slots. (In fact, it will become clear shortly that a nontrivial instance of the component retrieval

problem emerges only if at least one component type is assigned to at least two different feeder slots.) Components are gripped from a slot of the magazine rack and mounted on the PCB by a placement head. Coordination between the grip and place activities is done by a carousel, which performs many other functions as well. The carousel contains 12 heads, and it can simultaneously hold up to six components; see figure 1.

Suppose the machine is just about to place the i th component on the board. To this end, the board location where the component is to be placed is positioned at the so-called placement spot, and the carousel head containing the component (the current place station) is right above this spot. The carousel head then proceeds with the actual placement of the component on the board. After the placement has been completed, the worktable holding the PCB starts moving, until the board location where the next component is to be placed comes to rest at the placement spot. Diametrically opposed to the aforementioned carousel head is another head (the current grip station), which is positioned right above the so-called gripping spot. The grip station is ready to grip the $(i + 6)$ th component from the magazine rack as soon as the appropriate slot has been positioned at the gripping spot. Once this is done, the head proceeds with actually gripping the $(i + 6)$ th component from this feeder slot. After the gripping has been completed, the magazine rack starts to shift in order to position the slot from which the next component is to be retrieved at the gripping spot. Only after the i th component has been placed and the $(i + 6)$ th component has been gripped, is the carousel ready to rotate 30° clockwise to prepare for the placement and gripping of components $(i + 1)$ and $(i + 7)$, respectively.

Thus, between two consecutive place activities, the PCB table has to move until the board location where the second component is to be placed lies at the placement spot, and the carousel has to rotate 30° to position the next head right above the placement spot. Similarly, between two consecutive grip activities, the rack has to shift until the appropriate slot is at the gripping spot, and the carousel has to rotate to position the next head right above this spot. It is important to observe that placement operation i and gripping operation $(i + 6)$ do not have to be performed simultaneously, but are necessarily carried out between

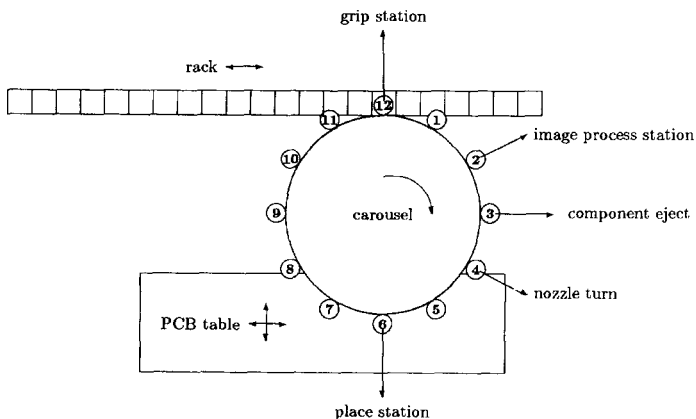


Figure 1. A Fuji CP placement machine.

the same two carousel rotations. Also, table, rack, and carousel movements may take place concurrently.

Clearly, the duration of rack movements depends on the distance between slots from which consecutive gripping operations are done. Therefore, even when the component placement sequence on the board and the magazine rack assignment of component tapes are given, minimizing the assembly makespan still involves decisions concerning the feeder slots from which each component should be retrieved. The corresponding optimization problem is known as the *component retrieval problem* (see Bard et al. 1994).

Of course, as mentioned earlier, the CRP involves nontrivial issues only if at least one component type is assigned to at least two different feeder slots. In our experience with a major industrial partner, we observed that such feeder duplication is common practice and sometimes improves productivity significantly (see Crama et al. forthcoming and Voogt 1993). Feeder duplication was also reported in several previous studies, including Ahmadi, Grotzinger, and Johnson (1988), Bard et al. (1994), Francis et al. (1994), and Klineciewicz and Rajan (1994).

3. The component retrieval problem as a PERT/CPM network model with design aspects

To facilitate our discussion, we present a PERT/CPM-like model of the CRP. To that end, we first need to introduce several assumptions and develop some notation. First, let $1, \dots, n$ denote the components that are to be mounted on the PCB, with the numbering reflecting their placement sequence on the board. With respect to the starting conditions of the assembly process, we assume that the feeder slot from which the first component will be retrieved is initially positioned below the grip station (currently occupied by carousel head 12) and that the PCB location where the first place activity will occur initially is positioned below the place station (currently occupied by carousel head 6). Furthermore, components 1–6 have been added as fictitious components, initially held by carousel heads 6–1, respectively; they are to be mounted at the same board location as component 7, an operation that can be performed in zero time. If we similarly assume that six fictitious and instantaneous grip activities are carried out at the end of the mounting process, then a situation has been constructed where exactly n grip activities and n place activities are required to assemble the board, with the i th grip and i th place activity occurring between the $(i - 1)$ th and i th carousel rotation.

Let us call a *retrieval plan* any feasible solution of CRP: thus, for every component $i = 1, \dots, n$, a retrieval plan determines from which feeder slot component i should be retrieved (a more formal definition will be stated later). As a first step toward modeling CRP, let us briefly explain how, for any given retrieval plan, the assembly makespan can be computed by classical PERT/CPM techniques. To this end, the *events* (i.e., moments in time) and *activities* (i.e., time durations) of table 1 are introduced. Events, activities, and precedence relations between activities can be represented by a PERT/CPM graph $D(S)$, where S is the retrieval plan under consideration, nodes and arcs correspond to events and activities, respectively, and arc lengths denote activity durations (see figure 2). We will refer to the

Table 1. Events and activities of the PERT/CPM graph $D(S)$.

g_i	= start of the i th grip activity	$(i = 1, \dots, n)$
p_i	= start of the i th place activity	$(i = 1, \dots, n)$
Δg_i	= duration of the i th grip activity	$(i = 1, \dots, n)$
Δp_i	= duration of the i th place activity	$(i = 1, \dots, n)$
Δm_i	= duration of the i th rack movement	$(i = 1, \dots, n - 1)$
Δt_i	= duration of the i th table movement	$(i = 1, \dots, n - 1)$
Δc_i	= duration of the i th carousel movement	$(i = 1, \dots, n - 1)$

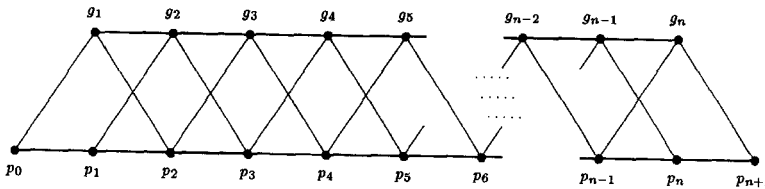


Figure 2. The PERT/CPM graph $D(S)$.

nodes as grip or place nodes, depending on the nature of the associated event. The resulting graph consists of n layers, where each layer i contains exactly one grip node g_i and one place node p_i ($i = 1, \dots, n$). To model the start and the end of the assembly process, it is convenient to add a source, indifferently denoted by p_0 and g_0 , and a sink, indifferently denoted by p_{n+1} and g_{n+1} . As is well-known, the makespan of the assembly process is equal to the length of a longest path in $D(S)$ from p to p_{n+1} . Computing a longest path calculation in such an acyclic and layered network can be done by forward dynamic programming in $O(n)$ time (see, e.g., Ahuja, Magnanti, and Orlin 1993).

To specify the arc lengths of $D(S)$, recall from section 2 that, between the start of two consecutive grip activities g_i and g_{i+1} ($i = 1, \dots, n - 1$), the following operations have to be performed: the i th grip activity, the i th carousel rotation, and the i th rack movement. Since the first precedes the other two, which may be carried out concurrently, it follows that the length of arc (g_i, g_{i+1}) equals $\Delta g_i + \max\{\Delta c_i, \Delta m_i\}$. On the other hand, the $(i + 1)$ th grip activity can start only when both the i th place activity and the i th carousel rotation are completed. Since these activities are carried out consecutively, it follows that the length of arc (p_i, g_{i+1}) equals $\Delta p_i + \Delta c_i$. Other arc lengths in $D(S)$ are defined in a similar fashion; see table 2.

In view of the preceding discussion, the component retrieval problem can now be modeled as follows. Consider the graph of figure 2. For each $i = 1, \dots, n$, we introduce a set of grip nodes G_i instead of only one grip node g_i , where each node of G_i refers to one of the slots containing the component type required for the i th grip activity. Figure 3 shows an example where the first two components (which may or may not be of the same type) can both be retrieved from two alternative feeder slots and all other components can be retrieved from only one such slot. Then, specifying a component retrieval plan, that is, a feasible solution of CRP, amounts to selecting exactly one grip node from each set G_i in such a way that the

Table 2. Arc lengths of $D(S)$ with $\Delta c_i = \Delta t_i = 0$ for $i = 0, n$ and $\Delta p_0 = 0$.

Arc	For	Length
(g_i, g_{i+1})	$i = 1, \dots, n - 1$	$\Delta g_i + \max\{\Delta c_i, \Delta m_i\}$
(p_i, p_{i+1})	$i = 0, \dots, n$	$\Delta p_i + \max\{\Delta c_i, \Delta t_i\}$
(g_i, p_{i+1})	$i = 1, \dots, n$	$\Delta g_i + \Delta c_i$
(p_i, g_{i+1})	$i = 0, \dots, n - 1$	$\Delta p_i + \Delta c_i$

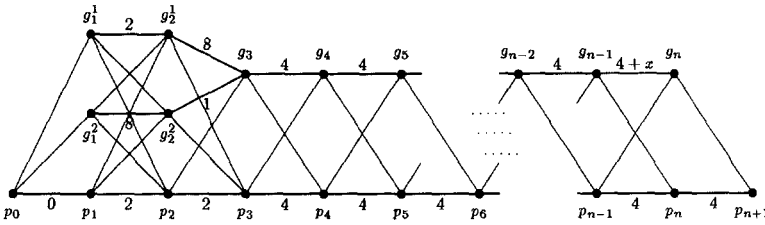


Figure 3. A counterexample of straightforward dynamic programming.

longest path in the subgraph induced by the selected nodes is as short as possible. We thus arrive at the following formalization of the CRP graphs and problems.

Definition 1 (CRP graph): A CRP graph $D = (V, A)$ is a layered directed graph on the node set $V = \cup_{i=0}^{n+1} L_i$, with the layers L_i being mutually disjoint sets. Moreover, $L_i = \{p_i\} \cup G_i$, where G_i is a nonempty set not containing p_i ($i = 1, \dots, n$) and $G_0 = G_{n+1} = \emptyset$. The set $\{p_0, \dots, p_{n+1}\}$ is referred to as the set of *place nodes*; all other nodes are called *grip nodes*. The arc set A is given by $A = \{(u, v) \mid u \in L_i, v \in L_{i+1} \text{ for some } i = 0, \dots, n\}$, with the length of the arc (u, v) being denoted by $d(u, v)$. The length function $d(\cdot, \cdot)$ satisfies

$$d(g_i, p_{i+1}) + d(p_i, g_{i+1}) \leq d(g_i, g_{i+1}) + d(p_i, p_{i+1}), \tag{1}$$

for all $g_i \in G_i, g_{i+1} \in G_{i+1}$ and $i = 1, \dots, n - 1$.

Note that the arc lengths displayed in table 2 trivially satisfy inequality (1). In the sequel (with the exception of Theorem 1), we will *not* make any explicit use of the specific lengths displayed in table 2, but rely on their property (1) instead. Thus, our definition of a CRP graph is more general than required to model the “real-world” CRP model. We view this as a virtue in several ways. First, CRP graphs may allow the formulation of the component retrieval problems associated with other types of placement machines than those explicitly considered here. Moreover, as we see in section 5, property (1) is sufficient to guarantee the efficient solvability of the component retrieval problem and, thus, provides a better understanding of why this problem is “easy”. Actually, we establish in section 6 that a generalized version of the CRP, in which (1) is no longer assumed to hold, is NP-hard.

Remark: The inequalities (1) are somewhat reminiscent of a matrix property studied in the literature under the name *Monge property* (see, e.g., Burkard, Klinz, and Rudolf 1996). The presence of this property is known to facilitate the solution of various combinatorial optimization problems such as the traveling salesman problem. We have not been able to establish any formal link between our results and previous work on the Monge property, however.

In the CRP graph model, *selections* provide a precise counterpart of the informal concept of a “retrieval plan.”

Definition 2 (Selection): A selection S in a CRP graph D is a set of grip nodes containing one grip node from each layer, that is, $|S \cap G_i| = 1$ for $i = 1, \dots, n$.

Definition 3 (Selection-induced subgraph): For any selection S in a CRP graph D , the selection-induced subgraph $D(S) = (V(S), A(S))$ is the subgraph of D induced by $S \cup \{p_0, \dots, p_{n+1}\}$. The length of the longest path in $D(S)$ is denoted $L[D(S)]$.

Since the length of a longest path in the subgraph induced by selection S is equal to the makespan of the PCB assembly process using the retrieval plan defined by S , we arrive at the following network version of the component retrieval problem.

Definition 4 (Component retrieval problem): Given a CRP graph D , the component retrieval problem (CRP) is to determine a selection S that minimizes $L[D(S)]$; namely, the length of a longest path in $D(S)$.

As mentioned before, the analysis and results in this paper will all apply to this network version of the component retrieval problem, not only to the special instances arising from the original application where arc lengths are defined as in table 2.

These definitions express that CRP is basically a PERT/CPM network problem with design aspects. Obviously, designs are restricted to selections in this case; that is, they must contain exactly one grip node per layer. As an alternative interpretation, the minimization of the makespan seems to indicate that all grip activities should be completed as early as possible. This, in turn, seems to suggest that a minimal makespan can be obtained by computing a shortest path from p_0 to p_{n+1} in the subgraph that is induced by these two place nodes and all grip nodes (the so-called grip graph). Unfortunately, the precedence relations induced by the place activities would be completely ignored in such an approach; a shortest path through the grip graph would specify an optimal selection only if, between each pair of grip nodes, the makespan (longest path length) that results from the interfering place activities (nodes) was taken into consideration as a lower-bounding side constraint. Therefore, the CRP can also be viewed as a shortest path problem with side constraints. Obviously, the side constraints are of a very specific nature here; viz., they result from longest path lengths induced by a single (place) path that is added to the (grip) graph under consideration. These two interpretations of the CRP seem interesting in their own right.

4. The CRP and forward dynamic programming

As briefly mentioned in the introduction, the component retrieval problem previously has been investigated by Bard et al. (1994), who proposed a forward dynamic programming scheme for its solution. Before we present our algorithm for CRP, we deem it necessary to explain why the approach proposed by Bard, Clayton and Feo *cannot* possibly lead to a correct algorithm for CRP.

Consider the CRP graph of figure 3. Let the arc lengths of (g_1^1, g_2^2) and (g_2^2, g_1^1) be “large” (say, larger than 10) and let all arcs (p_{i-1}, g_i) and (g_i, p_{i+1}) ($i = 1, \dots, n$) have length 0. The other arc lengths are as indicated in figure 3. Note that (g_{n-1}, g_n) has length $4 + x$. In the upcoming discussion, we will consider two possible values for x , $x = 0$ and $x = 5$, respectively.

Since $d(g_1^1, g_2^2)$ and $d(g_2^2, g_1^1)$ are “large,” the only candidate optimal selections are $S^1 = \{g_1^1, g_2^2, g_3, \dots, g_n\}$ and $S^2 = \{g_1^2, g_2^2, g_3, \dots, g_n\}$. Table 3 displays the longest path (length) in the corresponding selection-induced subgraphs, for $x = 0$ and $x = 5$, respectively. Observe that S^1 is optimal when $x = 0$, whereas S^2 is optimal when $x = 5$.

A simple forward dynamic programming scheme for CRP, with the set of grip nodes as state space, would be based on the following definition:

$$\phi(g_i) = \text{minimum length of a longest path from } p_0 \text{ to } g_i.$$

Note that $\phi(g_i) = 9 + 4(i - 3)$ for $3 \leq i \leq n - 1$, with the optimal predecessor of g_i being g_2^2 . However, when $x = 0$, then the unique optimal selection, i.e., S^1 , does *not* contain g_2^2 . Similarly, if the place nodes form the state space, that is,

$$\psi(p_i) = \text{minimum length of a longest path from } p_0 \text{ to } p_i,$$

then $\psi(p_i) = 6 + 4(i - 3)$ for $4 \leq i \leq n$, with the optimal predecessor of p_i being g_2^1 . Yet again, when $x = 5$, then g_2^1 is not part of the optimal selection S^2 . These observations clearly show that the principle of optimality does not hold in either case: to identify an optimal selection for the entire problem, it may be necessary to keep track of partial selections that are *non-optimal* up to certain layers.

In addition, the information required to track the first part of an optimal selection for the entire problem may be contained in arbitrarily remote parts of the graph, even as remote as the last grip arc. The conclusion is that simple forward dynamic programming does not necessarily identify optimal selections, not even if the recursion is equipped with a “look k

Table 3. Longest path (length) under different scenarios.

Longest paths	S^1	S^2
$x = 0$	$\{p_0, g_1^1, g_2^1, g_3, \dots, g_n, p_{n+1}\}$ length: $10 + 4(n - 3)$	$\{p_0, g_1^2, g_2^2, p_3, \dots, p_n, p_{n+1}\}$ length: $12 + 4(n - 3)$
$x = 5$	$\{p_0, g_1^1, g_2^1, g_3, \dots, g_n, p_{n+1}\}$ length: $15 + 4(n - 3)$	$\{p_0, g_1^2, g_2^2, g_3, \dots, g_n, p_{n+1}\}$ length: $14 + 4(n - 3)$

layers ahead or back” capability for constant k . To guarantee optimality, a more elaborate analysis and approach therefore seems to be required.

Let us stress that these negative conclusions directly affect the validity of the forward dynamic programming algorithm proposed by Bard et al. (1994). Indeed, the recursive formulation considered by these authors reads

$$f_{i+1}(k) = \min_{j \in Y_i} \{t_{jk}(i, i + 1) + f_i(j)\} \quad k \in Y_{i+1}, \quad i = 0, 1, \dots, n - 1, \quad (2)$$

where

$f_i(k)$ = minimum time required to grip the first i components given that the i th component is retrieved from magazine slot k ;

Y_i = set of magazine slots containing the component type required for the i th gripping activity;

$t_{jk}(i, i + 1)$ = elapsed time between completion of the i th gripping activity from slot j and the $(i + 1)$ th gripping activity from slot k .

Note that the interpretation of $f_i(k)$ coincides with that of $\phi(g_k)$ (see previously), if g_k is the node associated with feeder slot k in layer L_i of the CRP graph. What the example of this section shows (and what seems to have been overlooked by Bard et al. 1994) is that, in an *optimal* retrieval plan where the i th component is retrieved from slot k , the time required to grip the first i components may *strictly exceed* $f_i(k)$ for some i and k . An alternative way of understanding this conclusion is to realize that the time interval $t_{jk}(i, i + 1)$ is not univocally determined by j, k , and i (contrary to what its notation suggests) but actually depends on the sequence of grip activities prior to the i th one. This explains why recursion (2) does not lead to an easy algorithm for the CRP.

For the sake of completeness, let us add some more comments on a variant of the CRP that we encountered in a practical setting, which *can* be solved efficiently by forward dynamic programming. In Crama et al. (forthcoming), we describe an industrial case study in which the operating mode of the placement machine had been restricted as follows: for each $i = 1, \dots, n$, the start of the i th placement operation is required to coincide with the start of the $(i + 6)$ th gripping operation (in contrast to the description given in section 2). It is easy to see that recursion (2) is valid under this restriction. Indeed, the value of $t_{jk}(i, i + 1)$ can now be simply expressed as

$$t_{jk}(i, i + 1) = \max\{\Delta g_i + \Delta c_i, \Delta g_i + \Delta m_{jk}, \Delta p_i + \Delta c_i, \Delta p_i + \Delta t_i\}, \quad (3)$$

where Δm_{jk} is the duration of the rack movement from slot j to slot k , and the other notations have been previously defined. In particular, since the expression (3) depends only on j, k , and i , the assembly makespan can be computed in $O(m^2n)$ time, where $m = \max_{1 \leq i \leq n} |G_i|$, by solving recursion (2) (see Crama et al. forthcoming). In view of the relative simplicity of this procedure (as compared to the algorithm described in the next section), one may rightfully wonder to what extent the makespan of the selection that it delivers differs from the optimal makespan computed for the unrestricted machine.

More formally, for an arbitrary PCB, let S denote an optimal solution (viz., selection or retrieval plan) of the CRP and let $U = L[D(S)]$ denote the optimal assembly makespan of this PCB on a placement machine operating in unrestricted mode. Similarly, let S^{res} denote an optimal retrieval plan for the same PCB when the machine operates in restricted mode (viz., S^{res} is the solution of (2) when $t_{jk}(i, i + 1)$ is defined by (3)), denote by R the makespan of S^{res} on the restricted machine and denote by $H = L[D(S^{\text{res}})]$ the makespan of S^{res} on the unrestricted machine. Thus, R is the optimal assembly makespan for the restricted machine, whereas H (which stands for “heuristic”) is the makespan obtained on the unrestricted machine when we use the retrieval plan S^{res} rather than the optimal plan S . We are interested in the maximal value that can be achieved by each of the ratios R/U and H/U (notice that R/U measures the productivity loss that results from using the machine in restricted mode, whereas H/U measures the loss that results from using the suboptimal plan S^{res} rather than S). Under the (realistic) assumption that all data $\Delta g_i, \Delta p_i, \Delta c_i, \Delta t_i$, and Δm_{jk} are nonnegative, Theorem 1 holds.

Theorem 1: For every PCB, $H/U \leq R/U \leq 2$. Moreover, H/U and R/U can be made arbitrarily close to 2 for some PCBs.

Proof. Let us show that $H/U \leq R/U \leq 2$ for every PCB. First, notice that $H \leq R$, since the makespans H and R pertain to the same retrieval plan and the machine is clearly more efficient in unrestricted mode than in restricted mode. Therefore, we need to prove only that $R/U \leq 2$. Given an arbitrary PCB, consider the retrieval plan S that achieves the optimal makespan U on the unrestricted machine. Assume that S calls for placing components $1, \dots, n$ in this order, and for retrieving component i from slot $j(i)$ ($i = 1, \dots, n$ and $j(i) \in Y(i)$). Denote by M the makespan of S on the restricted machine. Then, we obtain successively

$$R \leq M \tag{4}$$

$$= \sum_{i=1}^n \max\{\Delta g_i + \Delta c_i, \Delta g_i + \Delta m_{j(i)j(i+1)}, \Delta p_i + \Delta c_i, \Delta p_i + \Delta t_i\} \tag{5}$$

$$\leq \sum_{i=1}^n \max\{\Delta g_i + \Delta c_i, \Delta g_i + \Delta m_{j(i)j(i+1)}\} + \sum_{i=1}^n \max\{\Delta p_i + \Delta c_i, \Delta p_i + \Delta t_i\} \tag{6}$$

$$\leq 2U. \tag{7}$$

Indeed, inequality (4) holds by optimality of R for the restricted machine, equality (5) follows from (3), and inequality (6) is trivial. As for inequality (7), observe that each sum in the right-hand side of (6) represents a lower bound on U , since each of them accounts for a sequence of operations—grip and place operations, respectively—that must necessarily be performed in succession. Thus, we have established the first part of the theorem.

We now provide a small example showing that H/U and R/U can be made arbitrarily close to 2. For the sake of simplicity, we assume in this example that the carousel of the machine only features two working heads and that it takes $|i - j|$ time units for the magazine rack to move from slot i to slot j , for every pair of slots i and j . (It would be an easy

matter to extend this example to account for the more complex features of real machines.) Three components, 1, 2, and 3, are to be placed in this order. Component 1 is contained in slot $2K$ of the magazine rack, component 2 is in slots $(K + 1)$ and $3K$, and component 3 is in slots 1 and $(3K + 1)$, where K is a given integer. Moreover, the worktable requires K time units to move from placement location 1 to placement location 2, and 1 time unit to move from location 2 to location 3. Each grip or place activity, and each rotation of the carousel, requires 1 time unit.

With these data, it is easy to check that the optimal retrieval plan S for the machine in unrestricted mode requires to grip component 2 from slot $3K$ and component 3 from slot $(3K + 1)$. This plan entails the following sequence of operations (operations listed in the same step are performed concurrently during the timespan indicated):

1. Grip component 1 in slot $2K$ (1 time unit).
2. Rotate the carousel, start moving the rack from slot $2K$ to slot $3K$ (for 1 time unit).
3. Place component 1, keep moving the rack toward slot $3K$ (for 1 time unit).
4. Move the rack to slot $3K$, start moving the table from location 1 to location 2 ($(K - 2)$ time units).
5. Grip component 2 from slot $3K$, keep moving the table toward location 2 (for 1 time unit).
6. Rotate the carousel, move the rack from slot $3K$ to slot $(3K + 1)$, move the table to location 2 (1 time unit).
7. Grip component 3 from slot $(3K + 1)$, place component 2 (1 time unit).
8. Rotate the carousel, move the table from location 2 to location 3 (1 time unit).
9. Place component 3 (1 time unit).

This sequence results in a makespan $U = (K + 6)$.

One would similarly verify that, for the machine in restricted mode, the optimal retrieval plan S^{res} consists in retrieving component 2 from slot $(K + 1)$ and component 3 from slot 1. The corresponding optimal makespan in restricted mode is equal to $R = (2K + 4)$ and identical to the makespan of S^{res} in unrestricted mode; that is, $H = (2K + 4)$. Therefore, when K goes to infinity, both R/U and H/U approach 2 as required. ■

Interestingly, it is also possible to prove that the makespan of the selection obtained by computing a shortest path in the grip graph (see the end of section 3) comes within a factor of 2 of the optimal CRP makespan and that this bound is tight. We omit the proof of this result.

In conclusion, all of the preceding comments underscore the need for an efficient and exact algorithm that takes into account all characteristic features of the component retrieval problem. Such an algorithm will be proposed in the next section.

5. A polynomial algorithm for the CRP

In this section, we consider a given CRP graph D and present a polynomial algorithm for the CRP as formulated in Definition 4. As explained in section 3, the optimal selection in D generally *cannot* be computed by solving for a shortest path in the subgraph induced by

the grip nodes of D , since the side constraints induced by the precedence relations of the interfering place activities would be completely ignored in that case. The general approach in this section is to model each of these side constraints as an arc between two grip nodes, with its length equal to the smallest longest path in D between these grip nodes. The optimal selection can then be retrieved by solving for the shortest path in this newly constructed graph, which will be denoted D_N . Since the arc lengths in D_N can be computed by a (polynomial and simple) forward dynamic programming approach and a shortest path in D_N can be computed likewise, our procedure can be thought of as a “two-phase” forward dynamic programming algorithm.

This section is built up as follows. First, a simplified version of the CRP will be considered, which can be solved by forward dynamic programming in polynomial time. The insights that have thus been obtained will then be used to arrive at a polynomial algorithm for the CRP itself. Application of the proposed algorithm to the numerical example of section 4 will conclude this section.

5.1. A simplified version of the CRP

Lemma 1: The length of the path $(p_0, p_1, \dots, p_{n+1})$ through the place nodes forms a lower bound on the optimal solution value of the CRP. ■

Proof. Straightforward.

This simple observation motivates our interest in the following problem.

CRP*

Input: A CRP graph D .

Question: Is there a selection S such that the path $(p_0, p_1, \dots, p_{n+1})$ through the place nodes is the longest path of $D(S)$?

Note that, if some selection S yields an affirmative answer to CRP*, then S is an optimal solution to the CRP (cf. Lemma 1). Physically, this simply means that, under the retrieval plan defined by S , the total duration of all place activities, carousel rotations, and table movements completely determines the assembly makespan; in other words, the place activities constitute a bottleneck for the makespan. Independent of its practical relevance, we will see in subsequent sections that CRP* provides a convenient stepping stone toward a complete solution of the CRP.

For $0 \leq i \leq j \leq n + 1$, let $L_P(i, j)$ be the length of the path $(p_i, p_{i+1}, \dots, p_j)$ from p_i to p_j through the place nodes. Similarly, for a selection $S = \{g_1, g_2, \dots, g_n\}$ and for $j - i \geq 2$, let $L_G(S, i, j)$ be the length of the path $(p_i, g_{i+1}, \dots, g_{j-1}, p_j)$ from p_i to p_j with all intermediate nodes in S .

Theorem 2: For every selection S , the path $(p_0, p_1, \dots, p_{n+1})$ is a longest path of $D(S)$ if and only if $L_G(S, i, j) \leq L_P(i, j)$ for all $i, j \in \{0, \dots, n + 1\}$ with $j - i \geq 2$.

Proof. If $L_G(S, i, j) > L_P(i, j)$ for some i, j , then the path $(p_0, p_1, \dots, p_i, g_{i+1}, \dots, g_{j-1}, p_j, \dots, p_{n+1})$ is longer than the path through the place nodes. On the other hand, the

inequalities in the theorem imply that the path through the place nodes will be at least as long as any path containing some grip nodes. ■

Theorem 2 motivates the introduction of a collection of *s-labels* associated with each selection *S*, which reflect the slack that *S* displays with respect to the necessary and sufficient conditions stated in the theorem.

Definition 5: For every selection *S* and every $j \in \{1, 2, \dots, n\}$, define

$$s(S, j) = \min_{0 \leq i \leq j-1} \{L_P(i, j + 1) - L_G(S, i, j + 1)\}. \tag{8}$$

We view label $s(S, j)$ as attached to the *j*th grip node of *S*. Theorem 2 can now be equivalently stated as follows.

Corollary 1: For every selection *S*, the path (p_0, \dots, p_{n+1}) is a longest path of $D(S)$ if and only if

$$s(S, j) \geq 0 \quad \text{for all } j \in \{1, 2, \dots, n\}. \tag{9}$$

The *s-labels* satisfy the following recursion.

Lemma 2: For every selection $S = \{g_1, g_2, \dots, g_n\}$ and every $j \in \{2, 3, \dots, n\}$,

$$s(S, j) = \min\{s(S, j - 1) + L_P(j, j + 1) + d(g_{j-1}, p_j) - d(g_{j-1}, g_j) - d(g_j, p_{j+1}), \\ L_P(j - 1, j + 1) - d(p_{j-1}, g_j) - d(g_j, p_{j+1})\}.$$

Proof. For each $i \in \{0, \dots, j - 2\}$, we can rewrite

$$L_P(i, j + 1) - L_G(S, i, j + 1) = [L_P(i, j) + L_P(j, j + 1)] \\ - [L_G(S, i, j) - d(g_{j-1}, p_j) + d(g_{j-1}, g_j) + d(g_j, p_{j+1})].$$

The validity of the lemma follows directly from this observation and from Definition 5. ■

Lemma 2 provides a recursive formulation of the *s-labels* associated with a given selection *S*. To solve CRP*, we now generalize the *s-labels* by introducing a label $s^*(g_j)$ attached to each grip node $g_j \in G_j$. The value of $s^*(g_j)$ is the largest value of $s(S, j)$ that can be attained by any selection *S* containing g_j and satisfying condition (9) up to layer $j - 1$. More precisely, we have the following definition.

Definition 6: For all $j \in \{1, 2, \dots, n\}$ and all $g_j \in G_j$, let $T(g_j)$ denote the set of selections *S* with (i) $g_j \in S$, and (ii) $s(S, i) \geq 0$ for all $i \in \{1, \dots, j - 1\}$. Then we define $s^*(g_j) = \max_{S \in T(g_j)} s(S, j)$.

As usual, we let $s^*(g_j) = -\infty$ when $T(g_j) = \emptyset$. Let us stress the following properties of the s^* -labels, which are direct consequences of Definition 6.

P₁: $-\infty < s^*(g_j) < 0$ if and only if $T(g_j) \neq \emptyset$ and $s(S, j) < 0$ for every selection $S \in T(g_j)$.

P₂: $s^*(g_j) \geq 0$ if and only if there exists a selection S with $g_j \in S$ and $s(S, i) \geq 0$ for all $i \in \{1, \dots, j\}$.

In particular, combining these properties with Corollary 1 renders Theorem 3.

Theorem 3: The answer to CRP* is affirmative if and only if $s^*(g_n) \geq 0$ for some node $g_n \in G_n$.

Similar to the s -labels, the s^* -labels can also be computed by dynamic programming (see Lemma 2).

Theorem 4: For all $j \in \{2, 3, \dots, n\}$ and for all $g_j \in G_j$,

$$s^*(g_j) = \max_{g_{j-1} \in G_{j-1}, s^*(g_{j-1}) \geq 0} \min\{s^*(g_{j-1}) + L_P(j, j + 1) + d(g_{j-1}, p_j) - d(g_{j-1}, g_j) - d(g_j, p_{j+1}), L_P(j - 1, j + 1) - d(p_{j-1}, g_j) - d(g_j, p_{j+1})\}. \quad (10)$$

Proof. Fix $j \in \{2, 3, \dots, n\}$ and $g_j \in G_j$. Denote the right-hand side of (10) by σ .

1. Assume first that $T(g_j) \neq \emptyset$. Then, by Definition 6, there exists an $S \in T(g_j)$ with $s^*(g_j) = s(S, j)$. If we write $S = \{g_1, g_2, \dots, g_n\}$, then it is clear that $S \in T(g_{j-1})$, so $s(S, j - 1) \leq s^*(g_{j-1})$. In addition, $s(S, j - 1) \geq 0$. Combining these two inequalities with Lemma 2 renders $-\infty < s^*(g_j) = s(S, j) \leq \sigma$.
2. Conversely, assume now that $\sigma > -\infty$, and let $g_{j-1} \in G_{j-1}$ attain the maximum in the definition of σ ; that is, $\sigma = \min\{a, b\}$ with $a = s^*(g_{j-1}) + L_P(j, j + 1) + d(g_{j-1}, p_j) - d(g_{j-1}, g_j) - d(g_j, p_{j+1})$ and $b = L_P(j - 1, j + 1) - d(p_{j-1}, g_j) - d(g_j, p_{j+1})$. By Definition 6, there exists a selection $S \in T(g_{j-1})$ with $s^*(g_{j-1}) = s(S, j - 1)$. Without loss of generality, we can assume that $g_j \in S$ (otherwise, substitute g_j for the j th grip node of S). Then, by Lemma 2, $s(S, j) = \sigma$. On the other hand, since $s^*(g_{j-1}) \geq 0$, we deduce that $S \in T(g_j)$ and, by Definition 6,

$$\sigma = s(S, j) \leq \max_{R \in T(g_j)} s(R, j) = s^*(g_j).$$

Taken together, 1 and 2 establish the theorem. ■

Theorem 4 implies that the s^* -labels can be computed in polynomial time, layer by layer. In view of Theorem 3, we thus have obtained a polynomial algorithm for the solution of CRP*. This algorithm can be implemented to run in $O(e)$ time, where e is the number of

arcs of D . Moreover, the proof of Theorem 4 establishes that, in addition to answering CRP*, we can also find a selection S with $s(S, j) \geq 0$ for all $0 \leq j \leq n$, if one exists. As a final remark, we observe that, up to this point, we have made no use of the properties of arc lengths recorded in Definition 1. In other words, Theorem 4 applies for arbitrary arc lengths.

5.2. Further properties of the s -labels

We have just described the role that the s -labels play in solving CRP*. In the next subsection, these ideas will be incorporated into an algorithm for the full-fledged component retrieval problem. To achieve this goal, we first need to understand some of the basic properties of the s -labels. These properties will now be recorded in a sequence of lemmas.

Lemma 3: For any $j \in \{1, 2, \dots, n\}$ and $g_j \in G_j$, let $S = \{g_1, g_2, \dots, g_n\}$ be a selection in $T(g_j)$. Consider $D(S)$. Then

1. The path (p_0, p_1, \dots, p_j) is a longest path from p_0 to p_j with length $L_P(0, j)$.
2. The path $(p_0, p_1, \dots, p_i, g_{i+1}, g_{i+2}, \dots, g_j)$ is a longest path from p_0 to g_j with length

$$L_P(0, j + 1) - s(S, j) - d(g_j, p_{j+1}), \tag{11}$$

where i is any index that realizes the minimum in the expression (8) defining $s(S, j)$.

Proof.

1. By Definition 6, $s(S, i) \geq 0$ for all $i \in \{1, \dots, j - 1\}$. The claim is now a straightforward extension of Corollary 1.
2. Let P_{g_j} be any longest path from p_0 to g_j and let $k = \max\{\ell \mid p_\ell \in P_{g_j}\}$. Since $0 \leq k \leq j - 1$, we have $S \in T(g_k)$, and hence (p_0, p_1, \dots, p_k) is a longest path from p_0 to p_k (cf. 1). Therefore, without loss of generality, we can assume that $P_{g_j} = (p_0, p_1, \dots, p_k, g_{k+1}, \dots, g_j)$. The length of P_{g_j} is now easily checked to be given by

$$L_P(0, j + 1) - [L_P(k, j + 1) - L_G(S, k, j + 1)] - d(g_j, p_{j+1}).$$

In view of (8), this expression is maximized when $L_P(k, j + 1) - L_G(S, k, j + 1) = s(S, j)$; that is, when $k = i$. Thus, we may indeed conclude that the path $(p_0, p_1, \dots, p_i, g_{i+1}, g_{i+2}, \dots, g_j)$ is a longest path from p_0 to g_j with length as stated in (11). ■

Next, let us consider what happens when, in a selection-induced subgraph $D(S)$, the longest path is *not* the path of place nodes $(p_0, p_1, \dots, p_{n+1})$; this is the only interesting case, since we know from the previous subsection how to handle yes-instances of CRP*. In such a case, we already know by Corollary 1 that $s(S, j)$ must be negative for some layer j . Let us consider the first such layer.

Lemma 4: For any selection $S = \{g_1, g_2, \dots, g_n\}$, let j be the smallest index in $\{1, 2, \dots, n\}$ with $s(S, j) < 0$. Then, in $D(S)$, every possible longest path from p_0 to p_{n+1} contains g_j .

Proof. Since every path from p_0 to p_{n+1} goes through either p_{j+1} or g_{j+1} , it suffices to show that g_j is contained in every longest path from p_0 to p_{j+1} and from p_0 to g_{j+1} . Since $S \in T(g_j)$ (by definition of j), the length of a longest path from p_0 to p_j (respectively, g_j) is given by Lemma 3. Thus, it suffices to show that

$$[L_P(0, j+1) - s(S, j) - d(g_j, p_{j+1})] + d(g_j, p_{j+1}) > L_P(0, j) + d(p_j, p_{j+1}) \quad (12)$$

(i.e., the longest path to p_{j+1} via g_j is longer than the longest path to p_{j+1} via p_j) and that

$$[L_P(0, j+1) - s(S, j) - d(g_j, p_{j+1})] + d(g_j, g_{j+1}) > L_P(0, j) + d(p_j, g_{j+1}) \quad (13)$$

(i.e., the longest path to g_{j+1} via g_j is longer than the longest path to g_{j+1} via p_j).

Now, (12) is trivially equivalent to the assumption that $s(S, j) < 0$. On the other hand, according to Definition 1,

$$d(g_j, g_{j+1}) + d(p_j, p_{j+1}) \geq d(g_j, p_{j+1}) + d(p_j, g_{j+1}). \quad (14)$$

Inequality (13) is then obtained by the addition of (14) to (12). ■

For an arbitrary selection $S = \{g_1, g_2, \dots, g_n\}$, Lemma 4 suggests that a longest path of $D(S)$ can be obtained by the following procedure. (Let us mention right away that this procedure is much more involved than necessary if its only purpose is to obtain a longest path of $D(S)$. The reason for considering it in this form is that it will rather naturally lead to an algorithm for CRP.) First, compute all labels $s(S, j)$ (e.g., layer by layer, as suggested by Lemma 2). If all s -labels are nonnegative, then we know that $(p_0, p_1, \dots, p_{n+1})$ is a longest path of $D(S)$. Otherwise, let

$$j = \min\{k \in \{1, \dots, n\} \mid s(S, k) < 0\}.$$

In view of Lemma 4, a longest path from p_0 to p_{n+1} in $D(S)$ can be obtained by concatenating a longest path from p_0 to g_j with a longest path from g_j to p_{n+1} . Accordingly, for any selection S , we call the first grip node $g_j \in S$ for which $s(S, j)$ is negative a *reset node* of $D(S)$. The term *reset* expresses that the computation of a longest path of $D(S)$ can be started anew from such a node. Now, by Lemma 3, a longest path from p_0 to g_j is readily available. So, we need to find only a longest path from g_j to p_{n+1} in $D(S)$. This subproblem clearly has the same structure as the problem we started with. More precisely, we can handle it as follows. We discard from $D(S)$ all layers with index $i \leq j$, except for g_j . Moreover, we decrease the length of both arcs (g_j, g_{j+1}) and (g_j, p_{j+1}) by $d(g_j, p_{j+1})$ (this is to account for the last term of (11); see (15)). Denote the new CRP graph thus constructed by $D_{g_j}(S)$. The observation we make now is that, as a consequence

of Lemma 3 and Lemma 4,

$$L[D(S)] = L_P(0, j + 1) - s(S, j) + L[D_{g_j}(S)] \tag{15}$$

(see Definition 3). The procedure just described can be applied iteratively until either g_n receives a nonnegative label or g_n becomes a reset node. In either case, let u_1, u_2, \dots, u_r denote the reset nodes sequentially identified in the process. Thus, for $k = 1, \dots, r, u_k$ is the reset node of $D_{u_{k-1}}(S)$ (where we let $u_0 \equiv p_0$). Denote by $s(S, u_{k-1}, u_k)$ the (negative) s -label attached to u_k in $D_{u_{k-1}}(S)$. The preceding discussion then can be summarized as follows.

Lemma 5: If u_k is the reset node of $D_{u_{k-1}}(S)$ for $k = 1, \dots, r$ and $D_{u_r}(S)$ has no reset node, then

$$L[D(S)] = L_P(0, n + 1) - \sum_{k=1}^r s(S, u_{k-1}, u_k). \tag{16}$$

Proof. This statement is a consequence of Lemma 3, Lemma 4, and the foregoing discussion. More precisely, let u_r lie in G_ℓ , where $1 \leq \ell \leq n$. Then, induction on (15) leads to

$$L[D(S)] = L_P(0, \ell + 1) - \sum_{k=1}^r s(S, u_{k-1}, u_k) + L[D_{u_r}(S)].$$

There are now two cases. If $\ell = n$, that is, the reset node u_r coincides with $g_n \in S$, then $L[D_{u_r}(S)] = 0$, from which (16) follows. Conversely, if g_n is not a reset node, then $L[D_{u_r}(S)] = L_P(\ell + 1, n + 1)$ (by Corollary 1), and (16) follows again. ■

Finally, consider two selections in $T(g_j)$, which are identical from layer j onward but have different s -label values at layer j . The next lemma states sufficient conditions for one of the selections to dominate the other one, as far as minimizing $L[D(S)]$ is concerned.

Lemma 6: For any $j \in \{1, 2, \dots, n\}$ and $g_j \in G_j$, let $S = \{g_1, g_2, \dots, g_n\}$ and $S' = \{g'_1, g'_2, \dots, g'_n\}$ be two selections in $T(g_j)$ with $g_i = g'_i$ for $i = j, \dots, n$. If $s(S, j) < s(S', j)$ and $s(S, j) < 0$, then $L[D(S')] < L[D(S)]$.

Proof. Let P be any longest path in $D(S')$. Then, P contains either p_j or g_j . In the first case, we can assume with no loss of generality that P contains p_0, \dots, p_j (cf. Lemma 3 sub 1), so that P is also a path in $D(S)$. But then Lemma 4 implies that P is not a longest path of $D(S)$, hence $L[D(S')] < L[D(S)]$. Conversely, if P contains g_j , then Lemma 3 sub 2 states that the subpath of P from p_0 to g_j has length $L_P(0, j + 1) - s(S', j) - d(g_j, p_{j+1})$, and that the longest path from p_0 to g_j in $D(S)$ has length $L_P(0, j + 1) - s(S, j) - d(g_j, p_{j+1})$. Since the latter is strictly larger than the former, the result follows. ■

5.3. The algorithm

Next we are going to show how Lemmas 5 and 6 can be combined to produce a polynomial time algorithm for the CRP. First, we reformulate and extend some of the notation introduced earlier. For every node $g_j \in G_j$ of D ($j = 0, 1, \dots, n - 1$), we denote by D_{g_j} the subgraph of D induced by the node set $\{g_j\} \cup \cup_{i=j+1}^{n+1} (\{p_i\} \cup G_i)$. The arc lengths in D_{g_j} are the same as in D , except that the length of each arc leaving g_j is decreased by $d(g_j, p_{j+1})$ for all $j \geq 1$. Note that D_{g_0} thus is identical to D . For each graph D_{g_j} , we can define s^* -labels as we did for graph D in Definition 6: the s^* -label attached to node g_k in D_{g_j} is denoted by $s^*(g_j, g_k)$ ($g_k \in \cup_{i=j+1}^n G_i$). In addition, $S_{g_j g_k}$ will refer to any selection that realizes the value of $s^*(g_j, g_k)$.

Before giving a more formal description of our algorithm, let us clarify the intuition behind it. Observe that, according to (16), the CRP can be seen as the problem of minimizing the expression $\sum_k -s(S, u_{k-1}, u_k)$ over all possible selections S . Let S be an optimal selection of the CRP graph D and let u_1, \dots, u_r be the corresponding sequence of reset nodes. By definition of the quantities $s(S, u_{k-1}, u_k)$, we have $s(S, u_0, u_1) = s(S, j)$ if u_1 is in layer j . Consider now the selection $S_{u_0 u_1}$, which is such that (by definition)

$$s(S_{u_0 u_1}, j) = s^*(u_0, u_1) = \max_{R \in T(u_1)} s(R, j)$$

and suppose that

$$s(S, j) < s(S_{u_0 u_1}, j).$$

Then, construct the selection S' that coincides with $S_{u_0 u_1}$ from layer 1 to layer j and that coincides with S from layer j to layer n . It follows directly from Lemma 6 that S' dominates S , and this contradicts the optimality of S . Thus, we have established that

$$s(S, j) = s^*(u_0, u_1),$$

or, equivalently,

$$s(S, u_0, u_1) = s^*(u_0, u_1).$$

By repeating this argument r times, we can derive that

$$-\sum_{k=1}^r s(S, u_{k-1}, u_k) = -\sum_{k=1}^r s^*(u_{k-1}, u_k). \tag{17}$$

In this way, we have reduced the CRP to the problem of minimizing the right-hand side of (17) over all possible choices of u_1, \dots, u_r , under the restriction that these nodes are the sequence of reset nodes associated with some selection.

We will now translate the latter problem into a shortest path problem in an auxiliary network D_N , where the length of each arc (g_j, g_k) is “essentially” equal to $-s^*(g_j, g_k)$. More precisely, the node set of D_N is $\{g_0\} \cup \bigcup_{i=1}^n G_i$. The arcs of D_N are all pairs of nodes of the form (g_j, g_k) , where $g_j \in G_j, g_k \in G_k$ and $0 \leq j < k \leq n$. The length of arc (g_j, g_k) is defined to be $w(g_j, g_k)$, where

$$\text{Case 1: } w(g_j, g_k) = -s^*(g_j, g_k) \text{ if } -\infty < s^*(g_j, g_k) < 0. \tag{18}$$

$$\text{Case 2: } w(g_j, g_k) = 0 \text{ if } s^*(g_j, g_k) \geq 0 \text{ and } k = n. \tag{19}$$

$$\text{Case 3: } w(g_j, g_k) = \infty \text{ otherwise.} \tag{20}$$

In view of Definition 6 and Theorem 4, Case 1 corresponds to a situation where g_k is a reset node in the subgraph of D_{g_j} induced by the selection S_{g_j, g_k} (see property P1 following Definition 6). Similarly, Case 2 occurs when there is no reset node in the subgraph induced by S_{g_j, g_n} up to and including layer n . Finally, in Case 3, any reset node of the subgraph induced by S_{g_j, g_k} lies either before g_k ($s^*(g_j, g_k) = -\infty$) or after g_k ($s^*(g_j, g_k) \geq 0$ and $k < n$).

Denote by $w(P)$ the length of a path P in D_N . For brevity, when we write *shortest path in D_N* we mean “shortest path in D_N from g_0 to some node in G_n , with respect to the length function w .” We are now finally ready for our next, and main, result. Note, however, that its proof is simply a formal generalization of the arguments presented already.

Theorem 5: The optimal value of the component retrieval problem on the graph D is equal to $L_P(0, n + 1) + w(P)$, where $w(P)$ is the length of a shortest path in D_N .

Proof.

1. Let S be an optimal selection for the CRP and let u_k denote the reset node of $D_{u_{k-1}}(S)$ ($k = 1, \dots, r; u_0 = g_0$). By Lemma 5, equation (16) holds. Now let $P' = \{u_0, u_1, \dots, u_r\}$ and consider any index $k \in \{1, \dots, r\}$. By the definition of reset nodes, $s(S, u_{k-1}, u_k) < 0$ and $s(S, u_{k-1}, u) \geq 0$ for all grip nodes u lying between u_{k-1} and u_k in S . Therefore, by Definition 6, we get $S \in T(u_k)$, where $T(u_k)$ is defined with respect to the graph $D_{u_{k-1}}$, and this implies that $-\infty < s(S, u_{k-1}, u_k) \leq \min\{0, s^*(u_{k-1}, u_k)\}$. If, for all $k = 1, \dots, r$, $w(u_{k-1}, u_k)$ is defined by either Case 1 or Case 2 (see (18) and (19)), then,

$$-s(S, u_{k-1}, u_k) \geq w(u_{k-1}, u_k) \quad \text{for } k = 1, \dots, r.$$

Combining these inequalities with (16) yields

$$L(D) \geq L_P(0, n + 1) + w(P'). \tag{21}$$

Now assume that $w(u_{k-1}, u_k)$ is defined by Case 3 (see (20)) for some k . In that case, $s^*(u_{k-1}, u_k) \geq 0$, or equivalently $s(S_{u_{k-1}, u_k}, u_{k-1}, u_k) \geq 0$ (note that $s^*(u_{k-1}, u_k) = -\infty$ has been ruled out earlier). Now let S' denote the selection that coincides with S from

p_0 to u_{k-1} and from u_k to p_{n+1} and that coincides with $S_{u_{k-1}u_k}$ from u_{k-1} to u_k . Apply Lemma 6 to the selections S and S' , both viewed as selections of $D_{u_{k-1}}$. This lemma implies that the longest path in the subgraph of $D_{u_{k-1}}$ induced by S' is shorter than the longest path in the subgraph induced by S , contradicting the optimality of S . As a result, the case that $w(u_{k-1}, u_k)$ is defined by (20) does not occur for arcs (u_{k-1}, u_k) on paths in D_N that are defined by the reset nodes of optimal selections.

2. Conversely, let $P = \{u_0, u_1, \dots, u_r\}$ be a shortest path in D_N , with $u_0 = g_0$ and $u_r \in G_n$. From point 1 it follows that $w(P) \leq w(P') < \infty$. Hence, for $k = 1, \dots, r-1$, the values $w(u_{k-1}, u_k)$ on P are all defined by (18), which means that u_k is a reset node in the subgraph of $D_{u_{k-1}}$ induced by the selection $S_{u_{k-1}u_k}$. Now consider the selection $S = \cup_{1 \leq k \leq r} S'_{u_{k-1}u_k}$, where $S'_{u_{k-1}u_k}$ is the set of nodes of $S_{u_{k-1}u_k}$ that lie between u_{k-1} and u_k . Lemma 5 implies that $L[D(S)] = L_P(0, n+1) + w(P)$ and hence

$$L(D) \leq L_P(0, n+1) + w(P). \quad (22)$$

From (21) and (22) we conclude that $L(D) \leq L_P(0, n+1) + w(P) \leq L_P(0, n+1) + w(P') \leq L(D)$. This establishes the result. ■

In summary, the component retrieval problem can be solved by the following algorithm.

Procedure SOLVE-CRP

begin

for all $j = 0, 1, \dots, n-1$ *and for all* $g_j \in G_j$ *do*

begin

set up the graph D_{g_j} ;

for all $k = j+1, \dots, n$ *and all* $g_k \in G_k$ *do*

begin

compute the label $s^*(g_j, g_k)$ of node g_k in D_{g_j} , and the corresponding

selection $S_{g_j g_k}$; define $w(g_j, g_k)$ according to (18), (19), and (20);

end

end

set up the graph D_N ;

compute a shortest path in D_N from g_0 to G_n with respect to the length function w ;

let $P = \{u_0, u_1, \dots, u_r\}$ denote this shortest path;

return the optimal selection $S = \cup_{1 \leq k \leq r} S'_{u_{k-1}u_k}$ with length $L(D) = L_P(0, n+1) + w(P)$

end

Theorem 6: Procedure SOLVE-CRP is correct and solves the component retrieval problem in $O(v e)$ time on a CRP graph D with v nodes and e arcs.

Proof. The correctness of procedure SOLVE-CRP follows from (the proof of) Theorem 5. As for its complexity, note that each execution of the loop “*for all j, for all g_j*” requires $O(e)$

time (by the comments following Theorem 4) and that this loop is executed $O(v)$ times. A shortest path in D_N can be found in $O(v^2)$ time since D_N is acyclic (see, e.g., Ahuja, Magnanti, and Orlin 1993).

The complexity of procedure SOLVE-CRP can be stated alternatively as follows. Let m be an upper bound on the number of feeders of each type; that is, $m = \max_{1 \leq i \leq n} |G_i|$. Then, $v = O(mn)$ and $e = O(m^2n)$, so that SOLVE-CRP runs in $O(m^3n^2)$ time. ■

5.4. Example

Now we illustrate the algorithm of the preceding subsection by applying it to the problem instance described in section 3, figure 3, with $n = 5$. Recall that $d(g_4, g_5) = 4 + x$ in this problem, with x being equal to either 0 or 5. The first phase yields the s^* -labels; the relevant values of these labels are listed in table 4.

The auxiliary graph D_N has the node set $\{g_0, g_1^1, g_1^2, g_2^1, g_2^2, g_3, g_4, g_5\}$; its relevant arcs are listed in table 5. If $x = 0$, the shortest path of D_N is (g_0, g_3, g_5) with a length of 2. Tracing back the predecessors in the third column of table 4 reveals the corresponding optimal selection $\{g_1^1, g_2^1, g_3, g_4, g_5\}$, which has a makespan of $16 + 2 = 18$ (see Theorem 5). On the other hand, if $x = 5$, then the shortest path in D_N is (g_0, g_2^2, g_5) with a length of 6.

Table 4. (Relevant) s^* -labels for the numerical example.

Grip node pair (g_i, g_j)	$s^*(g_i, g_j)$	Arg max in (10)
(g_0, g_1^1)	2	g_0
(g_0, g_1^2)	2	g_0
(g_0, g_2^1)	2	g_1^1
(g_0, g_2^2)	-4	g_1^2
(g_0, g_3)	-2	g_2^1
(g_2^1, g_3)	3	g_2^2
(g_2^2, g_4)	3	g_3
(g_3, g_4)	0	g_3
(g_2^2, g_5)	$3 - x$	g_4
(g_3, g_5)	$-x$	g_4

Table 5. (Relevant part of) graph D_N for the numerical example.

Arc	Length	Remark
(g_0, g_2^2)	4	
(g_0, g_3)	2	
(g_2^2, g_5)	2	Only if $x = 5$
(g_3, g_5)	5	Only if $x = 5$
(g_2^1, g_5)	0	Only if $x = 0$
(g_3, g_5)	0	Only if $x = 0$

The corresponding optimal selection reads $\{g_1^2, g_2^2, g_3, g_4, g_5\}$, which has a makespan of $16 + 6 = 22$. Note that these outcomes are consistent with the optimal selections that were reported in section 4.

6. An NP-hard generalization of the CRP

Our definition of the component retrieval problem includes condition (1) on the arc lengths of the CRP graph. This condition (which is satisfied by the PCB assembly application that originally motivated this work) has been explicitly used in the proof of Lemma 5 and, thus, was instrumental in deriving the polynomial-time algorithm presented in section 5. However, when we view the CRP as a more abstract network model (see the discussion following Definition 4), we may wonder whether condition (1) really is necessary to solve the problem efficiently. In this section, we answer this question by proving that the generalized version of the CRP obtained by removing condition (1) is NP-hard. Consider the following decision problem (the *generalized CRP*, or GCRP).

GCRP

Input: An integer β and a graph D satisfying the assumptions of Definition 1, except for (1).

Question: Is there a selection S such that the longest path in the selection-induced subgraph $D(S)$ has length no greater than β ?

Theorem 7: The GCRP is NP-complete, even if $|G_i| \leq 2$ for $i = 1, \dots, n$.

Proof. The GCRP is clearly in NP. Here we present a polynomial transformation from the NP-complete *even-odd partitioning* problem (EOP; see Garey and Johnson 1979) to the GCRP.

EOP

Input: N pairs of positive integers $I_i = \{x_{2i-1}, x_{2i}\}$ for $i = 1, \dots, N$.

Question: Is there an even-odd partition of $\{1, 2, \dots, 2N\}$, that is, a partition of $\{1, 2, \dots, 2N\}$ into disjoint subsets A and B with $|A \cap I_i| = |B \cap I_i| = 1$ for $i = 1, \dots, N$, and $\sum_{i \in A} x_i = \sum_{i \in B} x_i$?

Given an instance of EOP, we construct a graph D as in Definition 1, except that the arc lengths violate condition (1). We set $n = 4N$. For $k = 1, \dots, 4N$, each layer k contains three nodes; namely, one place node p_k and two grip nodes g_k^1 and g_k^2 . Layers $4i - 3$, $4i - 2$, $4i - 1$, and $4i$ are associated with pair I_i in the instance of EOP ($1 \leq i \leq N$). To define the arc lengths, we introduce three large numbers of different magnitudes:

$$Q = (N + 1) \cdot \max_{1 \leq i \leq 2N} \{x_i\}$$

$$K = (N + 1)Q$$

$$M = 4(N + 1)K.$$

The arc lengths in D are listed in table 6. (For notational convenience we indifferently denote the sink of D by p_{n+1} , g_{n+1}^1 , or g_{n+1}^2 .) Recall that arcs emanating from g_0 have length 0. Finally, we set $\beta = N(3K + Q) + \frac{1}{2} \sum_{i=1}^{2N} x_i$. This completely specifies an instance (D, β) of the GCRP. It remains to show that the instance of the GCRP obtained in this way and the original instance of EOP have the same answer. (Note that, as announced, the arc lengths do *not* satisfy condition (1), leaving the $P = NP$ question unanswered; indeed, for $i = 1, 2, \dots, N$, we have

$$d(g_{4i-2}^1, g_{4i-1}^1) + d(p_{4i-2}, p_{4i-1}) = 0 < 2Q = d(g_{4i-2}^1, p_{4i-1}) + d(p_{4i-2}, g_{4i-1}^1),$$

which contradicts condition (1).)

1. Suppose first that the instance of EOP has a positive answer and let (A, B) define an even-odd partition of $\{1, 2, \dots, 2N\}$. Without loss of generality we may assume that $A = \{2i - 1 \mid i = 1, \dots, N\}$ and $B = \{2i \mid i = 1, \dots, N\}$. Consider the selection S that contains $g_{4i-3}^1, g_{4i-2}^1, g_{4i-1}^1$, and g_{4i}^1 for i odd and $g_{4i-3}^2, g_{4i-2}^2, g_{4i-1}^2$, and g_{4i}^2 for i even ($i = 1, \dots, N$). We now claim that $L[D(S)] = \beta$, implying that the instance (D, β) has a positive answer.

Denote by D_i the subgraph of $D(S)$ induced by layers $4i - 3, 4i - 2, \dots, 4i + 1$, for every $i \in \{1, 2, \dots, N\}$. Two candidate longest paths in D_i are of the form

Table 6. Arc lengths of the GCRP instance (D, β) .

Arc	Length	For
(g_{4i-3}^1, g_{4i-2}^1)	$K + x_{2i-1}$	$i = 1, \dots, N$
(g_{4i-3}^2, g_{4i-2}^2)	$K + x_{2i}$	$i = 1, \dots, N$
(g_{4i-2}^1, g_{4i-1}^1)	0	$i = 1, \dots, N$
(g_{4i-2}^2, g_{4i-1}^2)	0	$i = 1, \dots, N$
(g_{4i-1}^1, g_{4i}^1)	$K + x_{2i}$	$i = 1, \dots, N$
(g_{4i-1}^2, g_{4i}^2)	$K + x_{2i-1}$	$i = 1, \dots, N$
(g_{4i}^1, g_{4i+1}^1)	K	$i = 1, \dots, N$
(g_{4i}^2, g_{4i+1}^2)	K	$i = 1, \dots, N$
(g_{4i}^1, g_{4i+1}^2)	K	$i = 1, \dots, N$
(g_{4i}^2, g_{4i+1}^1)	K	$i = 1, \dots, N$
(g_k^1, g_{k+1}^2)	M	$k \in \{4i - 3, 4i - 2, 4i - 1\}$
(g_k^2, g_{k+1}^1)	M	$k \in \{4i - 3, 4i - 2, 4i - 1\}$
(p_{4i-2}, p_{4i-1})	0	$i = 1, \dots, N$
Other place arcs ^a	K	
All cross arcs ^b	Q	

^aA place arc connects two place nodes.

^bA cross arc connects a grip and a place node.

$$\{p_{4i-3}, p_{4i-2}, g_{4i-1}^\delta, g_{4i}^\delta, g_{4i+1}^{3-\delta}\} \quad \text{and} \quad \{g_{4i-3}^\delta, g_{4i-2}^\delta, p_{4i-1}, p_{4i}, p_{4i+1}\}, \quad (23)$$

respectively, where $\delta = 1$ when i is odd and $\delta = 2$ otherwise. One of these paths has length $3K + Q + x_{2i-1}$ and the other has length $3K + Q + x_{2i}$. Furthermore, it is easily seen that all other paths in D_i are strictly shorter than the ones in (23). Using mathematical induction on N , then reveals that any longest path from g_0 to p_{4N+1} in $D(S)$ is the concatenation of paths in D_i of the types mentioned in (23) ($i = 1, 2, \dots, N$). Hence, the two candidate longest paths in $D(S)$ are

$$\begin{aligned} P_1 &= (g_0, g_1^1, g_2^1, p_3, p_4, p_5, p_6, g_7^2, g_8^2, g_9^1, g_{10}^1, p_{11}, p_{12}, p_{13}, p_{14}, g_{15}^2, g_{16}^2, g_{17}^1, \dots, p_{4N+1}) \\ &= \{g_0\} \cup \bigcup_{\substack{1 \leq i \leq N, \\ i \text{ odd}}} \{g_{4i-3}^1, g_{4i-2}^1, p_{4i-1}, p_{4i}\} \cup \bigcup_{\substack{1 \leq i \leq N, \\ i \text{ even}}} \{p_{4i-3}, p_{4i-2}, g_{4i-1}^2, g_{4i}^2\} \cup \{p_{4N+1}\} \end{aligned}$$

and

$$\begin{aligned} P_2 &= (g_0, p_1, p_2, g_3^1, g_4^1, g_5^2, g_6^2, p_7, p_8, p_9, p_{10}, g_{11}^1, g_{12}^1, g_{13}^2, g_{14}^2, p_{15}, p_{16}, p_{17}, \dots, p_{4N+1}) \\ &= \{g_0\} \cup \bigcup_{\substack{1 \leq i \leq N, \\ i \text{ odd}}} \{p_{4i-3}, p_{4i-2}, g_{4i-1}^1, g_{4i}^1\} \cup \bigcup_{\substack{1 \leq i \leq N, \\ i \text{ even}}} \{g_{4i-3}^2, g_{4i-2}^2, p_{4i-1}, p_{4i}\} \cup \{p_{4N+1}\}, \end{aligned}$$

with lengths $N(3K + Q) + \sum_{i=1}^N x_{2i-1} = N(3K + Q) + \sum_{i \in A} x_i = \beta$ and $N(3K + Q) + \sum_{i \in B} x_i = \beta$, respectively. Consequently, both candidate longest paths in fact are longest paths, and the answer to the GCRP instance (D, β) is like the answer to the EOP instance viz., affirmative.

2. Suppose next that the answer to the GCRP instance (D, β) is affirmative and let S be a selection of D with $L[D(S)] \leq \beta$. Since M is very large, $D(S)$ cannot contain any arc with length M . This means that, for each quadruple of layers $4i - 3, 4i - 2, 4i - 1$, and $4i$ ($i = 1, 2, \dots, N$), either all four grip nodes $g_{4i-3}^1, g_{4i-2}^1, g_{4i-1}^1$, and g_{4i}^1 or all four grip nodes $g_{4i-3}^2, g_{4i-2}^2, g_{4i-1}^2$, and g_{4i}^2 are in S . Therefore, S can be denoted by

$$S = \{g_0\} \cup \bigcup_{i=1}^N \{g_{4i-3}^{\alpha_i}, g_{4i-2}^{\alpha_i}, g_{4i-1}^{\alpha_i}, g_{4i}^{\alpha_i}\} \cup \{p_{4N+1}\}$$

with $\alpha_i \in \{1, 2\}$ ($i = 1, \dots, N$). Now consider the two paths

$$\begin{aligned} P'_1 &= (g_0, g_1^{\alpha_1}, g_2^{\alpha_1}, p_3, p_4, p_5, p_6, g_7^{\alpha_2}, g_8^{\alpha_2}, g_9^{\alpha_3}, g_{10}^{\alpha_3}, \\ &\quad p_{11}, p_{12}, p_{13}, p_{14}, g_{15}^{\alpha_4}, g_{16}^{\alpha_4}, g_{17}^{\alpha_5}, \dots, p_{4N+1}) \\ P'_2 &= (g_0, p_1, p_2, g_3^{\alpha_1}, g_4^{\alpha_1}, g_5^{\alpha_2}, g_6^{\alpha_2}, p_7, p_8, p_9, p_{10}, \\ &\quad g_{11}^{\alpha_3}, g_{12}^{\alpha_3}, g_{13}^{\alpha_4}, g_{14}^{\alpha_4}, p_{15}, p_{16}, p_{17}, \dots, p_{4N+1}). \end{aligned}$$

The lengths of these paths are $L(P'_1) = N(3K + Q) + \sum_{i \in A} x_i$ and $L(P'_2) = N(3K + Q) + \sum_{i \in B} x_i$, respectively, where (A, B) is a partition of $\{1, \dots, 2N\}$ with $|A \cap I_i| = |B \cap I_i| = 1$ for $i = 1, \dots, N$. Since both P'_1 and P'_2 have lengths that are no longer

than the longest path length in $D(S)$ and the latter in turn is no longer than β , it follows that $L(P'_1) \leq \beta$ and $L(P'_2) \leq \beta$. These observations, combined with the choice of $\beta = N(3K + Q) + \frac{1}{2} \sum_{i=1}^{2N} x_i$ renders $\sum_{i \in A} x_i = \sum_{i \in B} x_i$. Hence (A, B) is an even-odd partition of $\{1, \dots, 2N\}$, thus establishing that, like the GCRP instance (D, β) , the EOP instance allows for an affirmative answer. ■

7. Conclusions

The main contribution of this paper is a “two-phase” polynomial-time dynamic programming algorithm for the component retrieval problem, a problem that arises in the automated assembly of printed circuit boards. We have broadened the scope of our analysis by modeling the problem as a longest path minimization problem in a PERT/CPM-like network with design aspects. As an alternative interpretation, the problem can also be viewed as a shortest path problem with side constraints. Both interpretations have proven to be crucial in the development and description of the proposed solution algorithm. Finally, we have sharply delineated the complexity of the network model by proving that it becomes NP-hard when some special structure on the activity durations in the PERT/CPM network is absent.

Acknowledgments

The first author has been partially supported in the course of this research by AFOSR (grant F49620-93-1-0041), ONR (grants N00014-92-J-1375 and N00014-92-J-4083), and NATO (grant CRG 931531).

References

- 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 (Ed.), pp. 409–429, Kluwer Academic Publishers, Dordrecht, The Netherlands (1993).
- Ahmadi, J., Grotzinger, S., and Johnson, D., “Component Allocation and Partitioning for a Dual Delivery Placement Machine,” *Operations Research*, Vol. 36, No. 2, pp. 176–191 (1988).
- Ahuja, R.K., Magnanti, T.L., and Orlin, J.B., *Network Flows*, Prentice-Hall, Englewood Cliffs, NJ (1993).
- Ball, M.O. and Magazine, M.J., “Sequencing of Insertions in Printed Circuit Board Assembly,” *Operations Research*, Vol. 36, No. 2, pp. 192–201 (1988).
- Bard, J.F., Clayton, R.W., and Feo, T.A., “Machine Setup and Component Placement in Printed Circuit Board Assembly,” *The International Journal of Flexible Manufacturing Systems*, Vol. 6, No. 1, pp. 5–31 (1994).
- Burkard, R.E., Klinz, B., and Rudolf, R., “Perspectives of Monge Properties in Optimization,” *Discrete Applied Mathematics*, vol. 70, no. 2, pp. 95–161 (1996).
- Crama, Y., Flippo, O.E., van de Klundert, J.J., and Spieksma, F.C.R., “The Assembly of Printed Circuit Boards: A Case with Multiple Machines and Multiple Board Types,” *European Journal of Operational Research*, forthcoming.
- Crama, Y., Oerlemans, A.G., and Spieksma, F.C.R., *Production Planning in Automated Manufacturing*, 2nd ed., Springer-Verlag, Berlin, Germany (1996).

- Francis, R.L., Hamacher, H.W., Lee, C.-Y., and Yeralan, S., "Finding Placement Sequences and Bin Locations for Cartesian Robots," *IIE Transactions*, Vol. 26, No. 1, pp. 47-59 (1994).
- Garey, M.R. and Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, New York, NY (1979).
- Horak, T. and Francis, R.L., "Utilization of Machine Characteristics in PC Board Assembly," Working paper, Rutgers University, Newark, NJ (1995).
- Klincewicz, J.G. and Rajan, A., "Using GRASP to Solve the Component Grouping Problem," *Naval Research Logistics*, Vol. 41, pp. 893-912 (1994).
- Van Laarhoven, P.J.M. and Zijm, W.H.M., "Production Preparation and Numerical Control in PCB Assembly," *The International Journal of Flexible Manufacturing Systems*, Vol. 5, No. 3, pp. 187-207 (1993).
- Voogt, S., "Short Term Scheduling in PCB Assembly," Philips Report CTR 597-93-0106, Eindhoven, The Netherlands (1993).