# Application of Sequence-Dependent Traveling Salesman Problem in Printed Circuit Board Assembly

Ali Fuat Alkaya and Ekrem Duman

*Abstract*— **Optimization issues regarding the automated assembly of printed circuit boards attracted the interest of researchers for several decades. This is because even small gains in assembly time result in very important benefits in mass production. In this paper, the focus is on a particular placement machine type that has a rotational turret and a stationary component magazine. So far, this type of machine received little attention among the researchers. In this paper, the feeder configuration, placement sequencing, and assembly time minimization problems are formulated explicitly and completely (without simplifying assumptions) using nonlinear integer programming. In addition, the placement sequencing problem is shown to be a recently introduced new generalization of the traveling salesman problem (the sequence-dependent traveling salesman). These formulations show the complexity of the problems and the need for effective heuristic designs for solving them. We propose three heuristics that improve previously suggested solution methods and give comparable results when compared to simulated annealing that is a widely accepted good performing metaheuristic on combinatorial optimization problems. The heuristics are experimentally shown to improve previous methods significantly in assembly time that implies a huge economic benefit. The heuristics proposed could also be applied to other placement machines with similar operation principles.**

*Index Terms*— **Heuristic algorithms, optimization, quadratic programming, simulated annealing.**

## I. Introduction

OPTIMIZATION issues regarding the use of automated placement machines have attracted the interest of researchers for several decades. The automated placement machines are mainly used by telephone, computer, and TV set manufacturers to populate (assemble) electronic components on printed circuit boards (PCBs). Most electronic products contain PCBs as important components. PCBs are used extensively in a variety of products such as computers, calculators, robots, remote controllers, business telephones, cellular phones, and many electronic instruments.

Manuscript received May 27, 2012; revised January 26, 2013; accepted February 17, 2013. Date of publication April 18, 2013; date of current version May 29, 2013. This work was supported in part by the Scientific and Technological Research Council of Turkey under Project ID 108M198. Recommended for publication by Associate Editor S. J. Mason upon evaluation of reviewers' comments.

A. F. Alkaya is with the Department of Computer Engineering, Marmara University, Istanbul 34722, Turkey (e-mail: falkaya@marmara.edu.tr).

E. Duman is with the Industrial Engineering Department, Ozyegin University, Istanbul 34722, Turkey (e-mail: ekrem.duman@ozyegin.edu.tr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCPMT.2013.2252429

Presently, there are many types of placement machines available, such as sequential pick-and-place, dual-delivery, turret type (chip shooter), concurrent pick-and-place, and so on [1]. Various types of placement machines have different characteristics and restrictions [2]. Thus, the PCB production scheduling process is highly influenced by the type of placement machine used. For a complete survey on PCB assembly, see [3] and [4].

The PCB production process is an assembly line that involves solder paste, component placement, and solder reflow operations (soldering process is to adhere components on the PCB). A placement machine is very expensive and therefore, the assembly lines are typically designed such that the placement machine is the limiting resource or bottleneck, which is the key issue for assembly line optimization [5], [6]. Thus, for manufacturers to remain competitive in the growing PCB market, they must concentrate their efforts on improving the efficiency of their placement machines. Even small improvements in PCB assembly increase productivity and bring huge amounts of profits to the company.

The operation of these machines require two classes of decisions [7]. These are allocation of component types to feeder slots (also called as the feeder configuration problem) and determination of component placement sequence (also called as the placement sequencing problem). In some research, this list is extended but these two have great influence and hence importance in optimizing the PCB placement machines [3], [8]. They are interdependent, that is the solution of one affects the other and usually they are modeled as NP-complete problems [1], [7]. Hence, a solution aiming at achieving the optimum in both problems simultaneously is very difficult, if not impossible.

The placement sequencing problem is akin to traveling salesman problem (TSP) or its generalizations such as precedence constrained TSP [9]. In this problem, the aim is to find optimal placement order of components and return back to the starting point (to get ready for the next PCB) in a minimum amount of time. The feeder configuration problem is modeled as the quadratic assignment problem (QAP) in machines having a movable feeder magazine or the simple (linear) assignment problem [10] in some other machine types. In the feeder configuration problem, the goal is to assign component types to feeder slots therefore, total assembly time is minimized.

The literature for PCB assembly optimization problems is quite rich and it is possible to find lots of studies concerning the placement sequencing and feeder configuration problems [1]. However, most of these studies focus on other types of placement machines. For the machine type considered in this paper, there are a few directly related studies [11], [12].

In this paper, a particular placement machine having a rotational turret and a stationary component magazine, namely, the chip mounter, is undertaken. In a previous research, the operations of chip mounters are analyzed, definition of the placement sequencing and feeder configuration problems are given and solution procedures are suggested [11]. However it lacks a complete formulation of the problems and a comparison of proposed solution procedures with widely accepted metaheuristic approaches. Further improvements may be obtained by taking advantage of local search movements and operation principles of the machine.

In this paper, we give a nonlinear integer programming formulation of the placement sequencing problem and integer programming formulation of the feeder configuration problem arising from the operation principles of chip mounters. In these formulations every detail is considered and no simplifying assumptions are made. In addition, we show that the placement sequencing problem is a new generalization of the TSP introduced to the literature as the sequence dependent traveling salesman problem (SDTSP) [13]. In addition, the combined optimization problem is also formulated using a nonlinear integer programming model.

In addition, we propose three heuristics for improving the total assembly time. The performance of these heuristics improve former studies [11] by $> 9.5\%$ and in this production environment where the placement machines are mostly the bottleneck resources (they are very expensive), even a 1% improvement in assembly time has a significant impact for the manufacturer.

This paper is organized as follows. In Section II, we describe the operation principles of the analyzed machine, introduce the notation used for the formulations, and then give the formulations of the problems. In Section III, the previous research on the analyzed placement machine is summarized. In Section IV, we describe the details of the proposed heuristics. In Section V, we analyze the performance of the proposed heuristics on randomly generated problem instances and compare them with the former methods. In the last section, a summary of the heuristics and main conclusions are given together with some possible further research areas.

## II. Description of Operations and Problem Definition

The machine type considered in this research is the TDK brand and model RX-5A placement machine. This machine is usually encountered in assembly environments deploying surface mount technology. It is referred to as a chip mounter by the manufacturer and thus we prefer to use this name in this paper. To formulate the arising problem, we initially describe the operations of the machine. The machine is explained in detail in [11] but we provide a short explanation of the basic features of the machine for the sake of completeness. After giving the description of operations, we will introduce the notation to be used in the formulations and then give the formulations of the problems.

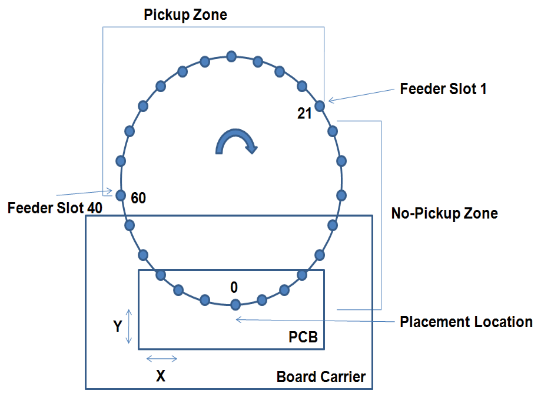

Fig. 1. Chip mounter.

### A. Description of Operations

Chip mounter has a rotational turret consisting of multiple heads that are responsible for the pickup of components from the component magazine and placing them on the PCB. The component magazine (also called the feeder mechanism) where the component tapes are installed is a circular structure behind the machine and it is stationary.

The rotational turret, with 72 heads, collects the components to its heads (one to each head) in the placement order, from the component tapes placed in the component magazine, while rotating in clockwise direction (Fig. 1). Even though the machine has actually 72 heads, in the figure we show a few for the sake of picture clarity. The head reaching the placement location over the board moves down and makes the placement to the precise location on the PCB that is prealigned with the placement position by the board carrier actions. Thus, each head makes the placement exactly at the same Cartesian coordinate and the alignment of the placement point is made by the board carrier through its simultaneous and independent movements (leading to a Chebyshev distance measure) in the $x$–$y$ plane.

These machines may handle component types of different weights (each head is equipped with three suction nozzles compatible with different weight categories). When any of the heads picks up a heavier component type, the rotation speed is reduced. That is, the speed of the turret is determined according to the heaviest component type carried. For the particular machine considered, there are four discrete speed values corresponding to different component weight categories. These are 0.20, 0.23, 0.33, and 0.40 s/5° (or, per rotational movement of distance one head) ordered from lightest to heaviest components. On the other hand, the board carrier movement speed in both $x$ and $y$ directions is 120 mm/s that could practically be taken as constant (i.e., no acceleration or deceleration).

The component tapes are mounted to a component magazine behind the machine and this whole system is stationary, that is, it does not move in any direction. If the placement heads are numbered with 0–71 in counterclockwise manner with number 0 being the placement head, up to 120 component tapes can be installed to the feeder slots along heads 21 and 60 that are in the pickup zone (there are 40 slots and based on their width one to three component types can be installed on a slot). The pickup times of the components from the magazine is completely determined by the order of their placement. Once the placement sequence is determined and the position of each component type in the magazine is entered to the computer, the component type that each head arriving at the placement location carries is known and fixed *a priori*. On the other hand, what component type each and every head should pickup is known. Then, during the rotational movement of the turret, the head(s) situating over the correct component tape move down and pick a component.

To summarize, a placement cycle of such a machine can be itemized as follows.

*Item 1:*

1) turret rotates and the next placement head comes over the PCB;
2) board carrier aligns the new placement point under the placement head;
3) placement heads rotate if necessary to align the suction nozzle carrying the component to be placed.

*Item 2:*

1) the placement head moves down, makes the placement, and moves up;
2) heads in the pickup zone that are above the appropriate component tapes move down and pickup components.

The actions under each item are performed concurrently, and there is a sequential order between items one and two.

There are two problems that must be defined, formulated, and solved for this machine. The first one is the placement sequencing problem and the other one is the feeder configuration problem. The assembly time minimization problem for the chip mounter machine is comprised of these two problems and its formulation will also be given.

## B. Notation

Before defining the problems, we give the notations that are used as follows: $N$ is the total number of components to be placed, $n$ is the number of component types, $K$ is the number of weight categories ($K = 4$ in our case), and $t_0$ is the component placement time (time for Item 2) This is a constant and unavoidable time for all component placements. In many cases, $t_0$ is small (in our case it is 0.05 s) and independent of the component placement sequence, therefore it may be dropped from the formulation without loss of generality. However, if we need to calculate the exact assembly time, we must add $N * t_0$ to the solutions of the formulations and examples given in the next subsections.

$N_k$ is the number of components in each weight category $k$, $k = 1, 2, \ldots, K, n_k$ is the number of component types in each weight category $k, k = 1, 2, \ldots, K, \sum_{k=1}^{K} n_k = n$, $R$ is the number of feeder slots, $H$ is the number of heads, and $npz$ is the number of heads in no pickup zone

$$c_{it} = \begin{cases} 1, & \text{if component } i \text{ is of component type } t \\ 0, & \text{otherwise.} \end{cases}$$

$g_{tk}$: group (weight category) matrix ($n \times K$) indicating the group for each component type

$$g_{tk} = \begin{cases} 1, & \text{if component type } t \text{ is in group } k \\ 0, & \text{otherwise.} \end{cases}$$

$\tau_k$ is the turret rotation time: time required for the next placement head to arrive over the PCB. Rotational turret rotates in different speed values that is determined according to the weight category ($k$) of the heaviest component type carried. (When $k = 1$, the turret rotation speed is the maximum, that is, turret rotation time is minimum), $S_k$ is the number of rotational steps of the turret in a speed category. This definition is needed for calculating a lower bound (LB), $p$ is the placement order of a component's placement on the board.

For the particular machine considered in this paper, $H = 71$ (excluding placement head), $R = 60$, and $npz = 20$. To express the placement sequence we need to define decision variable $x_{ip}$ that denotes the placement of component $i$ in placement order $p$

$$x_{ip} = \begin{cases} 1, & \text{if component } i \text{ is placed in } p\text{th placement} \\ & \text{order} \\ 0, & \text{otherwise.} \end{cases}$$

Decision variable $w_{\text{ijp}}$ is used for expressing the travel from component $i$ placed in $(p-1)$th placement order to component $j$ placed in $p$th placement order

$$w_{\text{ijp}} = \begin{cases} 1, & \text{if component } j \text{ is placed in } p\text{th placement} \\ & \text{order after component } i \text{ is placed in} \\ & (p-1)\text{th placement order} \\ 0, & \text{otherwise.} \end{cases}$$

On the other hand, to express the feeder configuration, we need to define decision variable $y_{tr}$ to state the status of assigning component type $t$ to feeder $r$

$$y_{tr} = \begin{cases} 1, & \text{if component type } t \text{ is stored in feeder } r \\ 0, & \text{otherwise} \end{cases}$$

where $r$ is the feeder number.

Component magazines are placed behind heads 21–60 where a feeder slot corresponds to each head. For simplifying the formulations below, we assume component magazines stand along heads 1–60 but no component types are assigned to feeder slots between 1 and 20 because they correspond to no pickup zone. Therefore, $r \in \{1, \ldots, 60\}$, however first 20 columns in the $y_{tr}$ array are always set to zero.

To give a better comprehension of the notation and operations of the machine, we give the simulation of a simplified chip mounter machine with eight heads, five feeder slots, and $npz = 2$. Assume the PCB to be assembled has six components identified with numbers one through six. Their component types and also group ID of each component type are shown in Table I. We have three component types and for avoiding confusion we identified them with numbers seven through nine. For simplicity, we have two groups of components with turret time values $\tau_1 = 0.20$ and $\tau_2 = 0.40$ s.

This table results in two matrices, $c_{it}$ and $g_{tk}$ that can be populated easily.

TABLE I
COMPONENTS, TYPES OF COMPONENTS, AND GROUPS OF TYPES FOR EXAMPLE PCB ASSEMBLY

| Component Number ($i$) | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Type of Component ($t$) | 8 | 9 | 7 | 7 | 8 | 7 |
| Group of Component Type ($k$) | 1 | 2 | 1 | 1 | 1 | 1 |

If the placement sequence is given as 1-6, then this results in the $x_{ip}$ matrix where only the diagonal elements are one, i.e., $x_{ip} = 1$ only for $i = p$.

In this small example, we have three feeder slots in the pickup zone. As a solution to the feeder configuration problem we assigned component type 7–9 to feeder slots 3–5, respectively.

The placement operations of the simplified chip mounter for assembling a small PCB is shown in Fig. 2. In mass production environments, the components arrive at the placement location continuously. As soon as the assembly of a board is completed the components on the heads are placed on the next board. Fig. 2(a) shows the empty PCB placed on the board carrier, the heads of the turret are carrying appropriate components and on the placement location a component of type 8 is ready for placement. Component locations with dashed lines mean components to be placed in the following steps whereas concrete lines mean already placed components. Type of the component is given in brackets and component number is given in the right bottom. Turret rotation time values after a placement is given in each subfigure.

As soon as the PCB is located on the board carrier, component 1 is placed by the placement head. Concurrently, the heads over feeder slots 3 and 4 move down and pickup components 4 and 5, respectively [Fig. 2(b)]. Then, the turret rotates one step and, simultaneously, the board carrier moves to align the placing position of component 2 and places component 2 [Fig. 2(c)]. After the placement in Fig. 2(c), the turret rotates one more step. Only during this rotation the turret time decreases $\tau_1 = 0.2$ s because all the components being carried are in group 1. During the placement of component 3, the heads over feeder slots 3–5 move down and pickup component 6 of this PCB and components 1 and 2 of the next PCB, respectively [Fig. 2(d)]. After three more placements, assembly of the current PCB is finished and the next PCB is started as in Fig. 2(a).

The assembly cost of this example is given after the placement sequencing problem is formulated in the next subsection where we address the placement sequencing problem by assuming that the feeder configuration is given.

## C. Placement Sequencing Problem

Given a feeder configuration, the objective of the placement sequencing problem is to find a placement sequence of the components therefore, the assembly process is completed in the shortest time possible. To express the placement sequencing problem, we need to define the time between the completion of consecutive placements at points $x$ and $y$. The time between consecutive placements at points $x$ and $y$ is

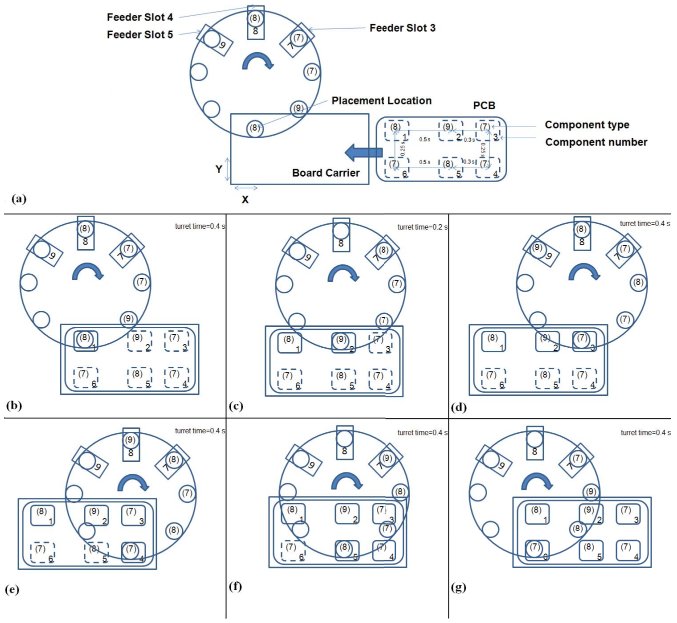

Fig. 2. Simulation of operations of simplified chip mounter. (a) Empty PCB. (b) Component 1 is placed and component 4 and 5 are picked up. (c) Component 2 is placed. (d) Component 3 is placed and component 6, 1 and 2 are picked up. (e) Component 4 is placed. (f) Component 5 is placed. (g) Component 6 is placed and component 3 is picked up.

defined by CF2

$$\text{CF2}(\tau_k, x, y) = \max(\tau_k, \text{CF1}(x, y)) \tag{1}$$

where $\text{CF1}(x, y)$ gives the travel cost from vertex $x$ to vertex $y$. It is typically a linear function that increases proportionally with distance. In this research, we calculate cost in time units, therefore we define $\text{CF1}(x, y)$ mathematically as follows:

$$\text{CF1}(x, y) = \frac{d(x, y)}{v} \tag{2}$$

where $v$ is a predefined speed value (speed of board carrier is 120 mm/s in our case) and $d(x, y)$ is the distance between points $x$ and $y$. Because of simultaneous and independent movements of the board carrier in the $x$–$y$ plane, $d(x, y)$ is defined as $\max\{|x_1 - x_2|, |y_1 - y_2|\}$, the Chebyshev metric, where $x_1$, $x_2$, and $y_1$, $y_2$ represent the $x$ and $y$-coordinates of points 1 and 2, respectively.

CF2 takes the maximum of two values because the analyzed machine has two simultaneously moving parts, both of which must finish their movement. One of these parts is the rotational turret and $\tau_k$ is the time to complete its rotation. Formally, we define turret time, $\tau_k$, as the turret rotation time required for the next placement head to arrive over the PCB. The turret time is also known as the free time because during this time interval the board carrier is free to move to any placement point that it can be without increasing the cost. On the other hand, the points at a Chebyshev distance of free time or less can be taken as equidistant points. When the component types to be placed are categorized according to their weight, we observe that there are four types of component types to be placed on the PCB. The speed of the turret is determined according to the heaviest component type carried. Hence $\tau_k$ takes four values such that $\tau_1 = 0.20$, $\tau_2 = 0.23$, $\tau_3 = 0.33$, and $\tau_4 = 0.40$ s.

$$\begin{tikzcd} 1 \arrow[r, "0.50"'] & 2 \arrow[r, "0.30"'] & 3 \arrow[r, "0.25"'] & 4 \arrow[r, "0.30"'] & 5 \arrow[r, "0.50"'] & 6 \arrow[r, "0.25"'] & 1 \end{tikzcd}$$

Fig. 3. Tour costs for small example.

The second part of the machine is the board carrier and the time for it to align the PCB under the placement head of the turret is calculated by CF1 as in (2).

If the rotational turret rotates at a unique speed value, then the problem turns out to be a classical TSP with Chebyshev distance measure. Thus, what makes the problem complicated is the varying rotational speed of the turret, resulting in a generalization of the TSP. Below, we show, how this difficult problem turns out to be the SDTSP introduced in [13].

Note a detail about indexes used in formulations below. In mass production environments, the components arrive at the placement location continuously and as soon as the assembly of a board is completed the components on the heads are placed on the next board. Hence, in the below formulations if an index exceeds its upper limit, it should be considered as if the counting of indexes restart from one. For example if we are talking about $(p-1)$th placement when $p$ is 1 and $N$ is 100, then $(p-1)$th placement means 100th placement on the previous board.

Then, cost of travel from component $i$ to component $j$ when component $j$ is assembled in $p$th placement order, $C_{\mathrm{ijp}}$ is

$$C_{\mathrm{ijp}} = w_{\mathrm{ijp}} \max \left\{ \begin{array}{l} CF1(i,j), \\ \\ \max\limits_{m=0}^{R-1} \left\{ \sum\limits_{l=1}^{N} \sum\limits_{t=1}^{n} \sum\limits_{k=1}^{K} \sum\limits_{u=m+1}^{R} \tau_k g_{tk} c_{lt} x_{l,p+m} y_{tu} \right\} \\ \\ \end{array} \right\} \tag{3}$$

where the first term in the outer max function is the time for board carrier to align the PCB under the placement head of the turret. The second term finds the maximum of turret time values of the following components that are carried by any of the heads, including component $j$. More explicitly, for calculating the placement time for component $j$ (placed in $p$th placement order), we need to answer the following question. Is the component to be placed $m$ steps later, (call it component $l$), currently carried by one of the heads or not? To answer this question, all feeder slots within indexes from $m$ to $R$ are scanned. If the component type of component $l$ is stored in one of these scanned slots, this means that component $l$ is carried by one of the heads and its turret time should be taken into consideration. Otherwise, component $l$ is not picked up yet, hence it should be neglected in the placement time calculation. This question should be answered for $R$ components to be placed after component $i$ and maximum turret time of those carried components effects the placement time. In short, the inner max function in (3) finds the maximum of turret time values of the following $R$ components that are carried by any of the heads.

Understanding of $C_{\mathrm{ijp}}$, consider the small example shown in Fig. 2, assuming that the CF1$(x, y)$ values for each consecutive travel in the tour is shown as follows in Fig. 3.

$C_{121}=max(0.50,\ max(0.4,\ 0.2,\ 0.2,\ 0.2,\ 0.0)) = 0.5$ seconds
$C_{232}=max(0.30,\ max(0.2,\ 0.2,\ 0.2,\ 0.0,\ 0.0)) = 0.3$ seconds
$C_{343}=max(0.25,\ max(0.2,\ 0.2,\ 0.2,\ 0.2,\ 0.4)) = 0.4$ seconds
$C_{454}=max(0.30,\ max(0.2,\ 0.2,\ 0.2,\ 0.4,\ 0.0)) = 0.4$ seconds
$C_{565}=max(0.50,\ max(0.2,\ 0.2,\ 0.4,\ 0.0,\ 0.0)) = 0.5$ seconds
$C_{616}=max(0.25,\ max(0.2,\ 0.4,\ 0.2,\ 0.0,\ 0.0)) = 0.4$ seconds

Fig. 4. $C_{\mathrm{ijp}}$ calculations for small example.

Then, the $C_{\mathrm{ijp}}$ values are calculated as shown in Fig. 4 and the total cost of assembly is calculated as 2.5 s.

CF2 is the generic formulation for calculating the time between the completion of two consecutive placements, whereas, $C_{\mathrm{ijp}}$ is the cost calculation formula to be used in an integer programming formulation and $C_{\mathrm{ijp}}$ clearly states and calculates turret time given in CF2.

## D. Feeder Configuration Problem

The objective of the feeder configuration problem is to find the optimum positioning of the component tapes within the magazine therefore, given a placement sequence, the total assembly time is minimized. The feeder configuration is of vital importance for optimizing the assembly time and we will show this by a small modification in the example shown in Fig. 2.

As another feeder configuration, assume that we assigned component types 7, 8, and 9 to feeder slots 5, 4, and 3, respectively. With this configuration, the total assembly cost reduces to 2.25 s. Therefore, it is important to make an effective feeder configuration to reduce total assembly time.

Next, we will give the formulation of the assembly time minimization problem that is the combination of placement sequencing and feeder configuration problems.

## E. Combined Problem and Formulations

Recall that, the ultimate objective of the optimization problem is to minimize the assembly time. That is, the objective is to find a placement sequence of the components and a feeder configuration therefore, the assembly process is completed in the shortest time possible. Hence, we build the following nonlinear integer programming that searches for optimum values of $x_{ip}$ and $y_{tr}$ variables simultaneously.

$C_{\mathrm{ijp}}$ is used for calculating the time between the completion of two consecutive placements and the objective is to minimize total assembly time. Therefore, to remove the nonlinearity in $C_{\mathrm{ijp}}$, we introduce $D_{\mathrm{ijp}}$ as the dominating factor in $C_{\mathrm{ijp}}$ definition in the following nonlinear integer programming formulation of the problem

$$\min \sum_{i=1}^{N} \sum_{\substack{j=1 \\ j \neq i}}^{N} \sum_{p=1}^{N} D_{\mathrm{ijp}} \tag{4}$$

s.t.

$$D_{\mathrm{ijp}} - w_{\mathrm{ijp}} CF1(i,j) \geq 0, \qquad i, j, p = 1, \ldots, N \tag{5}$$

$$D_{\mathrm{ijp}} - w_{\mathrm{ijp}} \sum_{l=1}^{N} \sum_{t=1}^{n} \sum_{k=1}^{K} \sum_{u=m+1}^{R} \tau_k g_{tk} c_{lt} x_{l,p+m} y_{tu} \geq 0$$

$$i, j, p = 1, \ldots, N, \quad m = 0, \ldots, R-1 \tag{6}$$

$$\sum_{p=1}^{N} x_{ip} = 1, \qquad i = 1, \ldots, N \tag{7}$$

$$\sum_{i=1}^{N} x_{ip} = 1, \qquad p = 1, \ldots, N \tag{8}$$

$$\sum_{i=1}^{N} w_{\text{ijp}} = x_{jp}, \qquad j, p = 1, \ldots, N \tag{9}$$

$$\sum_{j=1}^{N} w_{\text{ijp}} = x_{i,p-1}, \qquad i, p = 1, \ldots, N \tag{10}$$

$$\sum_{r=npz+1}^{R} y_{tr} = 1, \qquad t = 1, \ldots, n \tag{11}$$

$$\sum_{t=1}^{n} y_{tr} \le 1, \qquad r = npz+1, \ldots, npz+R \tag{12}$$

$$y_{tr} = 0, \qquad t = 1, \ldots, n, \quad r = 1, \ldots, npz, \tag{13}$$

$$x_{ip} \in \{0, 1\}, \qquad i, p = 1, \ldots, N \tag{14}$$

$$w_{\text{ijp}} \in \{0, 1\}, \qquad i, j, p = 1, \ldots, N \tag{15}$$

$$y_{tr} \in \{0, 1\}, \qquad t = 1, \ldots, n, \quad r = npz+1, \ldots, npz+R. \tag{16}$$

We introduce several constraint sets to the problem. Constraint set (5) imposes the requirement that placement time for component $j$ (component $j$ placed after component $i$ in $p$th placement order) cannot be less than the travel time from $i$ to $j$. Constraint sets (6) (representing $R$ different constraint sets) impose that placement time for component $j$ cannot be less than the turret time values associated with the incoming components for placement, if they are already picked up by the heads. Constraint (7) guarantees that each component is assigned to a placement order and constraint (8) guarantees that each placement order is occupied by only one component. There is no need for constraints that are used to guarantee subtour elimination, because placing each component in exactly one placement order is achieved by these constraints. Constraint (9) states that if component $j$ is placed in $p$th placement order, then exactly one component precedes $j$ in the travel sequence. Similarly (10) states that if component $i$ is placed in $(p-1)$th placement order then exactly one component proceeds it in the travel sequence.

Constraint (11) guarantees that each component type is assigned to a feeder slot and constraint (12) guarantees that each slot is occupied by at most one component type. Constraint (12) has a less than or equal to operator because number of component types may be less than number of feeder slots. We assign no component types to the slots corresponding to no pickup zone by constraint (13). Constraints (14)–(16) are the binary constraints for decision variables $x_{ip}$, $w_{\text{ijp}}$, and $y_{tr}$, respectively.

Having formulated the master problem, we can define the placement sequencing problem using one subset of constraints with $x_{ij}$ decision variables and the feeder configuration problem using another subset of constraints with $y_{tr}$ decision variables.

Given a feeder configuration, the placement sequencing problem can be easily formulated by using the same objective function and same set of constraints excluding (11)–(13), and (16). However, in the objective function and constraints (6) we must use the values of $y_{tr}$ variables given through feeder configuration.

Similarly, given a placement sequence, the feeder configuration problem can be easily formulated by using the same objective function and same set of constraints excluding (7)–(10), (14), and (15). In the objective function and constraints (5) and (6) we must use the values of $x_{ij}$ variables given through the placement sequence.

## III. Previous Work on Optimization of the Chip Mounter Machines

Even though it is stated that the placement sequencing problem is a hard one even to formulate, a solution methodology is developed in [11]. The developed methodology is based on the observation that mixing heavy and light components in the placement sequence is quite inefficient. Because, picking a component with a large turret time value causes a cost increase for a number of previous components even if they have smaller turret time values. Hence, to minimize the effect of components with larger turret time value to components with smaller turret time value, components having same (or close) turret time values should be visited successively. On the other hand, a Hamiltonian path (shortly named as path in the remaining text) must be built for each group and they must be connected by shortest connection. Therefore, given the feeder configuration, it seems advantageous to formulate separate TSPs for the groups of components in different weight categories. Accordingly, it seems to be a good idea to place all of the lighter components first and then the heavier ones (or, vice versa). This way, the $tt_k$ values corresponding to each weight category would be constant and it would be possible to use the TSP formulation to find the placement sequences within each weight category. This is the idea behind the assembly time minimization algorithm (ATMA) that is suggested in [11] for the solution of the placement sequencing problem. In ATMA, after finding the paths for each weight category, the paths are connected by shortest connection. Specifically, the group 2 component with shortest distance to the last component of group 1 is chosen as the starting point for path of group 2 and therefore, paths for group 1 and group 2 are connected. This process is repeated for group 2 and group 3 components and then for groups 3 and 4 components. Thus, a tour traversing all components is obtained (by a tour, we mean a set of paths connected one after another).

On the other hand, for the solution of the feeder configuration problem, the feeder arrangement procedure (FAP) is proposed in [11]. This procedure is based on the observation that, the above mathematical model implicitly requires the allocation of heavier component types closer to the PCB to make their discharging at the placement location quickly after a short travel. Therefore, FAP assigns the first $n_4$ slots to group 4, next $n_3$ slots to group 3, and so on. Even though, this approach seems to be a good way to solve the problem, it raises a new question: what should the internal arrangement within each group be? The idea in determining the internal arrangement of any group of components, is not to pickup any of them earlier than necessary. Otherwise a slowdown may occur one or more steps earlier. Therefore, Duman [11] reached the conclusion that a good solution is to determine the feeder configuration in the order of the first appearance of component types in the placement sequence. Being an ideal method for solving feeder configuration problem, we also used FAP in our research.

To summarize, in his solution approach, Duman [11] transformed the placement sequencing problem into four classical TSP problems. He achieved this by grouping components according to their weight categories. Solutions of these TSPs are found by using convex–hull and or–opt algorithms and they are connected by shortest connection. After the placement sequence is determined, feeder configuration problem is solved using FAP. Therefore, between the two optimization problems, the placement sequencing problem is solved first.

Another research toward optimizing the operations of chip mounter machines is [12]. In which, it is shown that reversing the placement process may bring improvements in terms of assembly time. They called it as inverse ATMA (iATMA) and placed the components by starting from heaviest-to-lightest. By coupling iATMA with other local search improvement techniques, $> 4.50\%$ improvement is obtained against ATMA [12].

## IV. Solution Methods

For solving the placement sequencing problem, grouping the components according to their turret time values, building separate paths for each group, and connecting these paths by shortest connection seems to be an effective way. Previous researches on the chip mounter machine use this idea to improve the throughput of the machine. However, further improvements can be obtained by effective local searches and replacements. In this paper, we propose two heuristics that bring further improvements to the grouping strategy used in [11] and [12].

These two heuristics are record-to-record travel with local exchange moves (RRTLEM) and adjust first point procedure (AFPP). AFPP is a simple but effective algorithm whereas RRTLEM is a local search heuristic.

On the other hand, mixing components belonging to different groups may sometimes produce more effective solutions. Based on this idea, a third heuristic called as postpone deviant (PD) is proposed. To compare the performance of these heuristics, we also implemented the simulated annealing (SA) metaheuristic as a benchmark algorithm.

### A. RRT With LEM (RRTLEM)

In TSP variants, for a local search, a general method is to apply exchanges of edges or nodes [14], [15]. Exchanging two edges is called as the two-opt move. For exchanging two nodes, we considered two alternatives; we can either insert

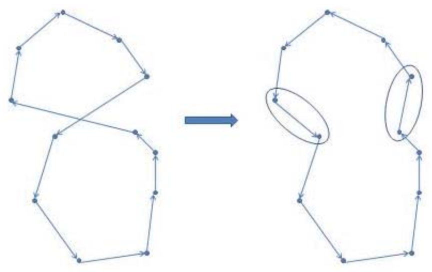
Fig. 5. Two-opt move.

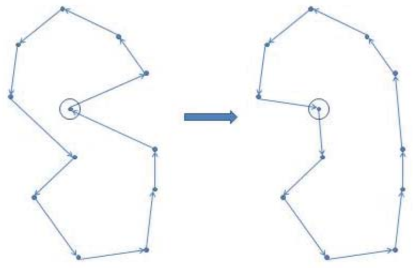
Fig. 6. Remove and reinsert move (special case of three-opt).

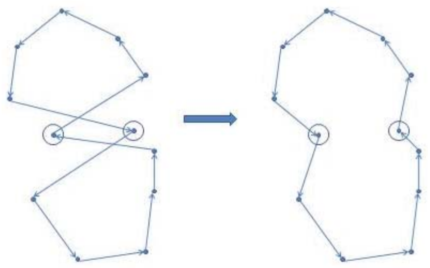
Fig. 7. Swap move (special case of four-opt).

a node into a different place on the tour, called as remove and reinsert move or we can exchange two nodes, called as swap move. Remove and reinsert and swap moves are actually special cases of three-opt and four-opt, respectively [16]–[18]. They are explicitly shown in Figs. 5–7.

RRTLEM is a hybrid of RRT and local search moves. RRT is, as a deterministic variant of SA, developed by Dueck and it is shown that the quality of the computational results obtained by RRT is better than SA on well-known TSPs [19] (Fig. 8). RRT starts by a generated initial solution, $s$. Record is defined as the cost of best solution (bs) observed so far. Deviation is defined as a predefined percentage of record. It is deterministic because, a neighbor solution $s$ replaces current solution only if its cost is less than Record+Deviation (other details of the algorithm are described in the following discussions).

We developed an improvement algorithm that combines RRT with local search moves. We call it RRTLEM. It is a general framework that consists of simple iterative statements (Fig. 9). In Fig. 10, an example is provided with remove and reinsert move for demonstrating how RRT concept is embedded in local search moves.

In remove and reinsert move with RRT, swap move with RRT, and two-opt Exchange move with RRT, we use the

```
s := GenerateInitialSolution()
bs := s
Record := f(bs)
Deviation := Record x p
while termination conditions not met
    s' := PickNextNeighbor(N(s))
    if f(s') < Record + Deviation
        s := s'
        if f(s') < Record
            bs := s'
            Record := f(s')
            Deviation := Record x p
        endif
    endif
endwhile
```

Fig. 8. RRT.

```
for each node i in the current solution cs
    savings := -∞
    for each edge j whose one end is in NL(i)
        obtain s’ by inserting node i between edge j
        if f(s’) < f(cs)
            cs := s’
            if f(s’) ≤ f(bs)
                bs := s’, dev := f(bs) x rate
            endif
            continue with the next node
        endif
        if f(s’) ≥ f(cs) and f(cs) - f(s’) ≥ savings
            store j as maximum saving edge (mse := j)
            savings := f(cs) - f(s’)
        endif
    endfor
    if f(cs) − savings ≤ f(bs) + dev and uphill moves allowed
        cs := the solution obtained by inserting node i between mse
    endif
endfor
```

Fig. 9. RRTLEM.

```
for each node i in the current solution cs
    savings := -∞
    for each edge j whose one end is in NL(i)
        obtain s’ by inserting node i between edge j
        if f(s’)<f(cs)
            cs := s’
            if f(s’)<=f(bs)
                bs := s’, dev := f(bs) x rate
            endif
            continue with the next node
        endif
        if f(s’)>=f(cs) and f(cs)-f(s’)>=savings
            store j as maximum saving edge (mse := j)
            savings := f(cs) - f(s’)
        endif
    endfor
    if f(cs)-savings <= f(bs)+dev and uphill moves allowed
        cs := the solution obtained by inserting node i between mse
    endif
endfor
```

Fig. 10. Remove and reinsert move with RRT.

solution obtained by ATMA as the initial solution. During the search process, current solution (cs) is the tour from which new solutions ($s$) are obtained and best solution (bs) is the tour whose cost is the best among the solutions created. Cost of solution $s$, $f(s)$, is the total assembly time of the PCB using the placement sequence stored in $s$. When uphill moves are allowed for a local search move, cs can be assigned a new solution whose cost is worse than itself but better than cost of bs plus dev, where dev is a predefined percentage (rate) of the cost of bs, and whenever a new bs is found, dev is also updated. Uphill moves create the opportunity for cs to escape from trapping local minima. After executing iterative statements, starting from the bs, we apply each local search move once more. The idea is inspired from [20]. In our algorithm, there are two parameters that must be fine-tuned to obtain best performance. These are the number of iterations, noi, and percentage value, rate, that is used to calculate the deviation.

Neighbor list is an important topic that should be explained. During local searches, it is more rational to try exchanges between nodes or edges that are away from each other at most a reasonable amount of distance. For example, for exchanging two nodes, it is not necessary to exchange the node at the most left and the most right. To avoid these unprofitable moves, we build neighbor lists for each node. In our implementation, we prefer to build neighbor lists consisting of nodes that can be reached within the turret time.

1. Obtain a placement sequence by using ATMA (Find Hamiltonian paths for each weight category. Call the path for weight category 1 as Path 1, and so on.)
2. For each component *i* in group 1,
   i. Modify the path of group 1 components such that placement starts from component *i*.
   ii. Connect Path 2 to the last point of modified Path 1 through the shortest connection. Rearrange Path 2 to make the connection point as the home city.
   iii. Repeat Step ii to connect Path 3 to the modified Path 2 and Path 4 to the modified Path 3.
3. The component that minimizes total assembly cost is chosen as starting point.

Fig. 11. AFPP.

## B. AFPP

Another heuristic we develop for the placement sequencing problem is AFPP that is a simple and effective algorithm. The idea behind AFPP is very simple. We initially build an initial tour by using ATMA. By a tour, we mean a set of paths connected one after another. Then for each point $i$ in path 1, we modify the path such that the tour starts from point $i$. This modification results in a change of last point of path 1 that causes a need for a reconnection of path 2 to the modified path 1 possibly through a new connection point. Therefore, path 2 is modified. Similarly, path 3 and path 4 are modified. Finally, the point that minimizes cost of the tour is chosen as the first point for the assembly. The procedure is shown in Fig. 11.

AFPP is a procedure that can be applied freely and easily in our case study. This is a consequence of continuous production of PCBs. As soon as assembly of a PCB is complete, assembly of another PCB starts. Therefore, we can choose any starting point for the assembly.

We expect AFPP to bring improvements because of two reasons. In the first place, AFPP looks for improvement opportunities that minimize total assembly cost where better (closer) connection points between groups exist. Secondly, starting the tour from a different point may cause more number of placements to be done within turret time. This is elaborated on more in the discussion section with examples. AFPP brings no change in cost calculation when considered under classical TSP constraints.

```
1. Sort the component groups from lightest to heaviest
2. For each group m
    • For each component i in group m
        • For each group n heavier than m
            • For each component j in group n
                • Temporarily insert i after j and if total
                  assembly cost decreases make this insertion
                  permanent
```

Fig. 12. PD heuristic.

```
Do
    1. assemblyTime = total assembly time of current
       placement sequence
    2. Call PD
    3. newAssemblyTime = total assembly time of current
       placement sequence
While newAssemblyTime ≠ assemblyTime
```

Fig. 13. EPD heuristic.

## C. PD

The general idea in the above heuristics is grouping components according to their turret time values, building a TSP tour for each group and placing these groups of components from lightest-to-heaviest or vice versa. However, in some cases this approach may produce inefficient placement sequences. Consider the following examples in which the placement sequence is built already (e.g., by ATMA).

Although placing a group of components, there may be some components whose placement location may be distant from their group. A component in a placement sequence is called as deviant when either travel to or from that location exceeds turret time. Placing these components among a heavier group of components may bring an improvement if its placement is completed within turret time.

As another example, consider a component whose removal from the sequence does not increase the total assembly cost, that is, direct travel from its preceding neighbor to its succeeding neighbor is completed within turret time. Placing such a component among another group of components may reduce total assembly time as it may connect two successively placed deviant components.

Placing lighter components among heavier ones may bring improvements whereas placing heavier ones among lighter components may bring inferiority in terms of total assembly time. This is because carrying a heavier component among lighter ones increases turret time and thus the placement time for a number of preceding lighter components. Hence, starting from lightest group of components, lighter components should be tried to be inserted among heavier components. PD heuristic is shown in Fig. 12. An extended version of the PD (EPD) heuristic can be defined as follows (Fig. 13).

The EPD calls the PD heuristic until no improvements are obtained in assembly time. It is expected that the EPD requires more time but gives better assembly time values. Analyses are given in Section V.

## D. Simulated Annealing

Another method we exploit for solving the combined problem is the SA metaheuristic. The fundamental idea is to allow moves resulting in solutions of worse quality than the current solution (uphill moves) to escape from local minima. The probability of doing such a move is decreased during the search. The algorithm starts by generating an initial solution, $s$, (either randomly or heuristically constructed) and by initializing the so-called temperature parameter $T$. Then, at each iteration a solution $s'$ in the neighborhood of $s$, $N(s)$, is randomly sampled and it is accepted as the new current solution based on $f(s)$, $f(s')$, and $T$, where $f(s)$ is the fitness value of solution $s$. $s'$ replaces $s$ if $f(s') < f(s)$ or, in case $f(s') \geq f(s)$, with a probability that is a function of $T$ and $f(s') - f(s)$. The probability is generally computed following the Boltzmann distribution $e^{(f(s')-f(s))/T}$ [21].

In our basic implementation of SA ($\text{SA}_{\text{basic}}$), a neighbor solution is obtained by exploiting pairwise exchanges in the placement sequence. The starting solution is chosen randomly. In the initial stages, $T$ is high and pairwise exchanges with small deterioration are likely to be accepted so as to avoid being prematurely trapped in a local optimum. As pairwise exchanges continue, $T$ approaches zero and most uphill moves are rejected. Other details of the SA implementation are inspired from [22], as their design has a demonstrated superiority on another combinatorial optimization problem in PCB assembly, namely the QAP.

Later, during the course of this research we implement an improved version of SA ($\text{SA}_{\text{improved}}$). $\text{SA}_{\text{improved}}$ makes use of both the neighbor information and move operators that RRTLEM uses. That is, in $\text{SA}_{\text{improved}}$, rather than any random two components or edges, two close components or edges are selected and used in a move operator (same way in RRTLEM). In addition, at each iteration, the $\text{SA}_{\text{improved}}$ randomly chooses one of the three operators that RRTLEM uses. The parameters of both SA implementations are defined as follows.

*Initial Temperature ($T$):* The larger this value, the more the inferior exchanges encouraged. In numerical computations, $T$ is set to either 100 or 1000. $T$ equaling 1000 corresponds roughly to accepting 10% worse solutions with a probability of 0.75, whereas $T$ equaling 100 corresponds roughly to accepting 10% worse solutions with a probability of 0.05.

*Temperature Decrease Ratio ($a$):* After a predetermined number of iterations, $T$ is set to $T/a$ (i.e., $T := T/a$). When $a$ is large, the temperature decrease is faster and the acceptance of inferior exchanges becomes less likely at a greater rate. In numerical computations, $a$ is set to either 1.1 or 1.5.

*Number of Iterations at Each Temperature Setting ($P$):* Greater values of $P$ correspond to slower cooling, that is, more exchanges occurring when there is a greater likelihood of inferior exchanges being accepted. In numerical computations, $P$ is set to either 5 or 20.

*Increase Ratio in Iteration Number at Each Setting ($b$):* After a predetermined number of iterations, $P$ is set to $P * b$ (i.e., $P := P * b$). In numerical computations, $b$ is set to either 1.1 or 1.5.

*Maximum Number of Exchanges (*nox*):* $N^3$ is the investigated maximum number of iterations where $N$ is the number of components in our case. That is, after $N^3$ pairwise exchanges, the algorithm stops no matter what the temperature is. Each parameter setting is also tested for $(N^3/3)$ number of iterations. This is to see the effect of the number of iterations on the performance of the solution procedure.

## V. Results of Numerical Investigation and Discussion

To demonstrate the complexity of the problem, we try to obtain the exact solutions of reasonably sized instances. The above-given combined problem formulation is implemented in GAMS environment using DICOPT solver. However, even problems with 30 components could not be solved because of memory constraints of the computer with eight gigabytes of memory. This is because the number of variables scale quadratically with the number of components and the number of equations scale cubically with the number of component types. For example, for the combined problem on small chip mounter and PCB example given in the previous sections, the number of variables is 51 and number of equations to be solved is 1653. For a board with 30 components and seven component types to be assembled on a real chip mounter, number of variables and number of equations increase to 1320 and a value $> 1.62$ million, respectively. For a moderate PCB with 100 components and 50 component types, the above numbers increase to 13 000 and a value $> 60$ million. Therefore, obtaining the exact result in a reasonable time seems to be impossible with the current computing technologies.

However, to compare the performances of the heuristics developed, we have another important and effective criterion, namely the LB, which arises because of the nature of chip mounters. LB is defined in the next subsection.

The above discussion also reminds us that effective heuristics should be developed for solving the assembly time minimization problem. In the following subsections, after defining the benchmark criteria, the performance analyses of the proposed heuristics on large problems are presented.

### A. Benchmark Criteria

To compare the performance of our algorithms, we implement a PCB data generator. The methodology for this generator is the same as the one used in [11]. The total number of components to be placed on a board, $N$, is set to 100, where the number of component types in group 2–4 are determined uniformly between 1 and 5, where each of them is placed one, two or three times (with equal probabilities) on the PCB. The board that these components are placed is assumed to have dimensions of 250 mm by 300 mm and it is assured that no two components are placed on the same coordinate. The generator can generate two types of PCB data: 1) homogeneous boards and 2) structured boards. By homogeneous board, we mean that while determining the location of any component, coordinates are selected purely random; whereas by structured boards, we put a restriction on locations of components of heavier types, i.e., groups 2–4. An investigation on real PCB manufacturing systems reveals that the components in the heavier groups appear to be placed more closely. Thus, structured board types more or less represent the real PCBs. But for the sake of a complete study, we will investigate the performance of our algorithm on both PCB types. The data generator generates 100 instances for both PCB types to be used as input files. Here, any value appearing in a comparison table is the average of 100 PCBs, unless stated otherwise. The tests are performed on a machine with Intel Core2 CPU at 2.0 GHz with 2-GB RAM.

The basic and most important comparison criterion in PCB placement machines is the total assembly time of a given PCB. As it is difficult to find the optimum PCB assembly time, the solution quality of the proposed heuristics can be assessed through a LB that is discussed below.

If we assume the feeder can be arranged ideally for all weight groups then the number of placements in a speed category $k$ $(SC_k)$ can be determined easily as follows:

$$S_1 = \max\{0, (N_1 - (20 + n_4 + n_3))\} \tag{17}$$

$$S_2 = \max\{0, (N_1 + N_2 - S_1 - (20 + n_4))\} \tag{18}$$

$$S_3 = \max\{0, (N_1 + N_2 + N_3 - S_1 - S_2 - 20)\} \tag{19}$$

$$S_4 = N - S_1 - S_2 - S_3. \tag{20}$$

These $S_k$ calculations are valid for ATMA and their detailed explanation and justification can be found in [11]. The logic behind these formulas is that before all of the light components are discharged at the placement location, the successing heads start picking up heavier components that cause an earlier slow down of the turret. The $S_k$ calculations regarding iATMA are different (as the placement order is reversed to heavier and then the lighter components) and a discussion on their calculation can be found in [12].

Returning back to the LB discussion, if we further assume the $x$–$y$ movements of the board carrier can be completed within the allowed turret time (i.e., $\text{CF1}(x, y) \leq \tau_k$ for all consecutively placed components) then, the sum of the number of placements in a speed category $k$ times the corresponding $\tau_k$ values plus $N \cdot t_0$ (total constant placement time) constitutes a LB for the PCB assembly time. Mathematically, LB is

$$\text{LB} = N \cdot t_0 + \sum_{k=1}^{K} \tau_k S_k. \tag{21}$$

A third criterion for comparing the heuristics is the run time. However, we should state that in real life conditions these problems are solved in advance and thus run time should be given less consideration. Therefore, the assembly time should be considered as the basic comparison operator. The running times provided are solely for information purposes.

### B. Parameter Fine Tuning

We mention about two parameters of RRTLEM heuristic that must be fine-tuned to obtain the best performance. These are the number of iterations, noi, and percentage value, rate, which is used to limit the deviation. After making a large number of tests, we decide that rate parameter should be equal to 1.4% and noi parameter should be equal to 400 for both homogeneous and structured boards.

On the other hand, SA has some parameters that must be fine-tuned for the computational experiments. After large number of tests, the following values are observed to have best performance for both $SA_{basic}$ and $SA_{improved}$; $T = 100$, $P = 5$, $a = 1.1$, and $b = 1.1$. The results of those tests are shown in the Appendix. The nox is a parameter whose higher value is expected to give better results and this is the case in fine tuning experiments. However, for the SA implementations to be comparable with other heuristics in terms of running time, we provide the results with nox $= N^3/3$.

### *C. Results*

We compared the performance of our heuristics and that of SA variants using the deviation from the LB. LB indicates the minimum time that a board can be assembled. As formulated, it depends on turret time values multiplied by the number of steps taken in each turret time. If we can make the assembly of a PCB in LB time, then the optimum is definitely reached and the machine operates in full performance. The deviation columns in the following tables represent the deviation from the LB.

Each SA variant is run 50 times for each input file and minimum assembly time of these 50 replications is recorded. From tables, each run time regarding SA variants is the average of 50 replications. On the other hand, the consecutively applied heuristics will be indicated by a+between them. For example, ATMA+AFPP+RRTLEM indicate that initial solution is obtained by ATMA, then AFPP is use to improve the solution that is followed by RRTLEM. As EPD performs better than PD, we give the results of EPD only. The results for structured boards are shown in Table II and results for homogeneous boards are shown in Table III.

Specifically, EPD is 12.02% and 10.16% away from the LB when applied after ATMA and iATMA, respectively. RRTLEM shows better performance than EPD, yet requires a larger run time. Whenever RRTLEM and/or EPD are/is applied, they/it definitely bring/s improvement in terms of total assembly time.

AFPP demonstrates a deviation of about 10% for both ATMA and iATMA in a very small run time. This is because of its capability to search for more components to be placed within turret time and improvement opportunities that minimize total assembly cost where better (closer) connection points between groups exist. Next subsection elaborates this discussion with examples.

As can be observed, combining any two heuristics brings further improvements. When all of them are applied successively after ATMA, the deviation from the LB decreases down to 7.89%. Combining the superiority of iATMA with these heuristics, the assembly time is minimized further and 7.66% deviation is obtained. However, it is interesting to observe that applying iATMA, AFPP, and RRTLEM successively shows inferior performance when compared to iATMA+RRTLEM. This is a possible result of local search methods because they do not guarantee to end with a better final solution when started with a better initial solution. It is also easily observed that even though $SA_{basic}$ takes much more run time, it demonstrates the worst performance. On the other hand, $SA_{improved}$ shows a comparable performance with best set of heuristics using the same running time. Therefore, we can conclude that an effective design can be built by taking problem specific measures into account. Additionally, the variance among the 50 runs of $SA_{basic}$ and $SA_{improved}$ are 1.90

TABLE II
PERFORMANCE COMPARISON OF HEURISTICS ON STRUCTURED BOARDS

| Heuristic | Ass. Time (s) | Runtime (s) | Deviation |
|---|---|---|---|
| ATMA | 35.83 | 0.51 | 14.88% |
| $SA_{Basic}$ | 38.72 | 22.87 | 24.13% |
| $SA_{Improved}$ | 33.76 | 22.56 | 8.23% |
| ATMA + EPD | 34.94 | 9.46 | 12.02% |
| ATMA + RRTLEM | 34.15 | 11.37 | 9.49% |
| ATMA + AFPP | 34.38 | 0.55 | 10.22% |
| ATMA + AFPP + RRTLEM | 33.91 | 11.36 | 8.70% |
| ATMA + AFPP + EPD | 34.07 | 9.33 | 9.23% |
| ATMA + RRTLEM + EPD | 33.77 | 20.55 | 8.27% |
| ATMA + AFPP + RRTLEM + EPD | 33.65 | 19.70 | 7.89% |
| iATMA | 34.58 | 0.52 | 10.85% |
| iATMA + EPD | 34.36 | 8.10 | 10.16% |
| iATMA + RRTLEM | 33.75 | 10.44 | 8.20% |
| iATMA + AFPP | 34.34 | 0.50 | 10.09% |
| iATMA + AFPP + RRTLEM | 33.77 | 10.43 | 8.26% |
| iATMA + AFPP + EPD | 34.15 | 7.56 | 9.48% |
| iATMA + RRTLEM + EPD | 33.57 | 18.25 | 7.62% |
| iATMA + AFPP + RRTLEM + EPD | 33.58 | 17.34 | 7.66% |

TABLE III
PERFORMANCE COMPARISON OF HEURISTICS ON HOMOGENEOUS BOARDS

| Heuristic | Ass. Time (s) | Runtime (s) | Deviation |
|---|---|---|---|
| ATMA | 40.55 | 0.53 | 30.51% |
| $SA_{Basic}$ | 40.88 | 22.54 | 31.56% |
| $SA_{Improved}$ | 35.49 | 22.32 | 14.24% |
| ATMA + EPD | 38.51 | 10.80 | 23.93% |
| ATMA + RRTLEM | 38.63 | 11.74 | 24.34% |
| ATMA + AFPP | 38.62 | 0.54 | 24.29% |
| ATMA + AFPP + RRTLEM | 37.97 | 11.55 | 22.20% |
| ATMA + AFPP + EPD | 37.11 | 10.56 | 19.45% |
| ATMA + RRTLEM + EPD | 37.11 | 20.92 | 19.45% |
| ATMA + AFPP + RRTLEM + EPD | 36.70 | 20.95 | 18.11% |
| iATMA | 40.20 | 0.50 | 29.40% |
| iATMA + EPD | 38.63 | 10.35 | 24.31% |
| iATMA + RRTLEM | 38.95 | 10.83 | 25.34% |
| iATMA + AFPP | 39.21 | 0.51 | 26.18% |
| iATMA + AFPP + RRTLEM | 38.48 | 10.55 | 23.84% |
| iATMA + AFPP + EPD | 37.67 | 10.43 | 21.25% |
| iATMA + RRTLEM + EPD | 37.44 | 20.05 | 20.51% |
| iATMA + AFPP + RRTLEM + EPD | 37.07 | 19.47 | 19.32% |

TABLE IV
ASSEMBLY TIME DEVIATION FROM LB FOR ATMA + AFPP + RRTLEM + EPD

| $(N)$ | ATMA+AFPP+RRTLEM+EPD | | $SA_{improved}$ | |
|---|---|---|---|---|
| | Hom. | Str. | Hom. | Str. |
| 100 | 18.73% | 7.42% | 13.50% | 7.90% |
| 200 | 9.06% | 2.97% | 15.28% | 8.35% |
| 300 | 5.48% | 1.16% | 15.92% | 11.35% |
| 400 | 3.78% | 0.90% | 17.55% | 11.43% |

and 0.10 s, respectively, which shows that the obtained solution is robust.

When performance of the algorithms on homogeneous boards is analyzed, best performance is displayed by $SA_{improved}$ where the variance for the $SA_{basic}$ and $SA_{improved}$ are 1.25 and 0.18 s, respectively. Applying the proposed heuristics after iATMA shows inferior results to coupling them with ATMA. This result is mostly based on the poor performance of iATMA on homogeneous boards. An extended discussion on this topic is given in [12]. Another possible cause is the nature of local search heuristics that they do not guarantee to end with a better final solution when started with a better initial solution.

Another set of results we will discuss is about the deviation from LB for the more promising heuristics for varying number of components. In Table IV we show the assembly time deviation from LB for the best performing heuristics combination: ATMA+AFPP+RRTLEM+EPD. The tests are performed on both homogeneous and structured boards.

In Table IV, we observe that deviation from the LB for ATMA+AFPP+RRTLEM+EPD converges to zero as number of components on a board increase. This result could be expected because as the number of components ($N$) increases, the average $x$–$y$ distances between component pairs decreases and it will be easier to arrange the placement sequence therefore, the board carrier movements can be completed within the turret time, so in LB time. The larger deviations in homogeneous boards are because of the heavier components are more spreaded (as compared with the structured ones) and therefore it might not be possible to place most of them within turret time. These values are an average of ten PCBs and we observe in several structured PCB instances with 400 components that the assembly plan of ATMA+AFPP+RRTLEM+EPD is only 0.40 s away from LB. Therefore, very good values, if not optimal, are obtained. On the other hand, SA displays a worse performance as $N$ increases. This is because as the solution space is larger, the temperature decreases to 0 before the algorithm converges to the optimum solution. A search on the best set of parameter values may overcome this drawback.

## D. Discussion on AFPP

The performance of AFPP within a small amount of time when compared with others is worth attention. As before, success of AFPP is due to two reasons. First, AFPP looks for improvement opportunities that minimize total assembly cost where better (closer) connection points exist. Consider the following example shown in Figs. 14 and 15.

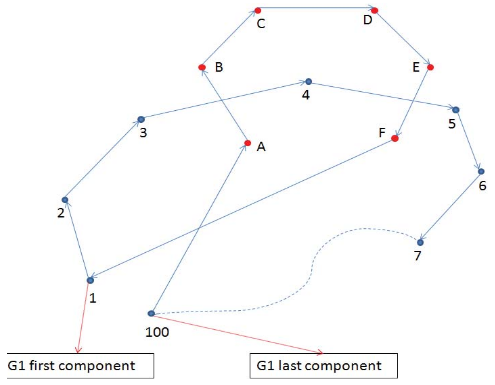

Fig. 14. Example depicting ATMA solution.

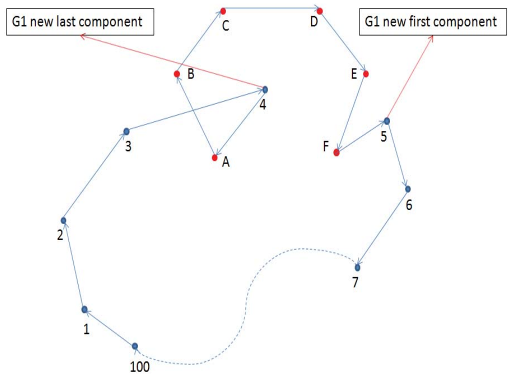

Fig. 15. First example showing improvement of AFPP.

$$A \xrightarrow[0.5]{} B \xrightarrow[0.40]{} C \xrightarrow[0.30]{} D \xrightarrow[0.20]{} E \xrightarrow[0.40]{} F \xrightarrow[0.20]{} G \xrightarrow[0.30]{} H \xrightarrow[0.10]{} I \xrightarrow[0.20]{} J \xrightarrow[0.30]{} A$$

Fig. 16. Second example showing improvement of AFPP.

Here, for simplicity assume we have two groups of components and number of group 1 components is 100 (labeled numerically) whereas number of group 2 components is six (labeled with characters from A to F). The solution shown in Fig. 14 is given by ATMA and the assembly starts from the southwest component, component 1. After the route of group 1 components is finished, assembly continues with the group 2 component closest to the last component of group 1, component 100. When assembly of group 2 components is complete, assembly of a board starts over from component 1. It is seen that distance between connection points increases the assembly cost.

On the contrary, we observe small distances between connection points in Fig. 15. This cost reduction opportunity is caught by AFPP while trying new starting points for the assembly.

Secondly, starting the tour from a different point may cause more number of placements to be made within turret time. As an example, consider the following TSP route for group 1 components with two turret time values (Fig. 16). Assume $\tau_1$ is 0.20 s, $\tau_2$ is 0.40 s, and as the parameter in CF2, $\tau_1$ is used for the first five placements whereas $\tau_2$ is used for the rest. That is, after placing these group 1 components group 2 components are picked up and therefore turret rotation slows down.

In addition, CF1$(x, y)$ values are given between each pair of components (the numbers below the arrows). If the starting point for the tour is selected as $A$, then total assembly cost for group 1 components is calculated as 3.4 s. On the other hand,

TABLE V
RESULTS OF EXPERIMENTAL RUNS

| Parameter Value Sets | | | | | Min | | Avg. | | Variance | |
|---|---|---|---|---|---|---|---|---|---|---|
| $T$ | $P$ | $a$ | $b$ | nox | Hom. | Str. | Hom. | Str. | Hom. | Str. |
| 100 | 5 | 1.1 | 1.1 | $N^3$ | **33.84** | 32.97 | **34.43** | **33.33** | **0.09** | **0.04** |
| 100 | 5 | 1.1 | 1.1 | $N^3/3$ | 35.26 | 33.88 | 36.08 | 34.62 | 0.18 | 0.11 |
| 100 | 5 | 1.1 | 1.5 | $N^3$ | 52.57 | 49.90 | 54.60 | 52.05 | 0.60 | 0.83 |
| 100 | 5 | 1.1 | 1.5 | $N^3/3$ | 53.59 | 51.38 | 56.17 | 54.14 | 0.93 | 1.12 |
| 100 | 5 | 1.5 | 1.1 | $N^3$ | 34.21 | 33.16 | 35.34 | 34.19 | 0.42 | 0.42 |
| 100 | 5 | 1.5 | 1.1 | $N^3/3$ | 34.27 | 33.22 | 35.39 | 34.22 | 0.40 | 0.40 |
| 100 | 5 | 1.5 | 1.5 | $N^3$ | 33.85 | **32.94** | 34.54 | 33.43 | 0.15 | 0.08 |
| 100 | 5 | 1.5 | 1.5 | $N^3/3$ | 34.12 | 33.11 | 34.84 | 33.62 | 0.16 | 0.09 |
| 100 | 20 | 1.1 | 1.1 | $N^3$ | 35.34 | 34.10 | 36.04 | 34.64 | 0.12 | 0.07 |
| 100 | 20 | 1.1 | 1.1 | $N^3/3$ | 40.84 | 38.12 | 42.47 | 39.47 | 0.44 | 0.32 |
| 100 | 20 | 1.1 | 1.5 | $N^3$ | 53.21 | 50.88 | 55.23 | 53.10 | 0.71 | 0.84 |
| 100 | 20 | 1.1 | 1.5 | $N^3/3$ | 54.25 | 51.93 | 56.82 | 54.77 | 1.05 | 1.20 |
| 100 | 20 | 1.5 | 1.1 | $N^3$ | 34.24 | 33.18 | 35.34 | 34.13 | 0.40 | 0.36 |
| 100 | 20 | 1.5 | 1.1 | $N^3/3$ | 34.22 | 33.20 | 35.36 | 34.18 | 0.41 | 0.33 |
| 100 | 20 | 1.5 | 1.5 | $N^3$ | 33.92 | 33.02 | 34.48 | 33.41 | 0.09 | 0.05 |
| 100 | 20 | 1.5 | 1.5 | $N^3/3$ | 35.81 | 34.24 | 36.79 | 34.99 | 0.25 | 0.11 |
| 1000 | 5 | 1.1 | 1.1 | $N^3$ | 38.79 | 36.63 | 39.96 | 37.53 | 0.24 | 0.15 |
| 1000 | 5 | 1.1 | 1.1 | $N^3/3$ | 46.00 | 42.93 | 47.94 | 44.75 | 0.61 | 0.62 |
| 1000 | 5 | 1.1 | 1.5 | $N^3$ | 54.50 | 52.76 | 57.33 | 55.36 | 0.89 | 1.10 |
| 1000 | 5 | 1.1 | 1.5 | $N^3/3$ | 55.66 | 53.35 | 58.26 | 56.63 | 1.13 | 1.48 |
| 1000 | 5 | 1.5 | 1.1 | $N^3$ | 34.11 | 33.14 | 35.36 | 34.16 | 0.45 | 0.42 |
| 1000 | 5 | 1.5 | 1.1 | $N^3/3$ | 34.19 | 33.26 | 35.43 | 34.23 | 0.40 | 0.40 |
| 1000 | 5 | 1.5 | 1.5 | $N^3$ | 34.84 | 33.54 | 35.50 | 34.00 | 0.11 | 0.06 |
| 1000 | 5 | 1.5 | 1.5 | $N^3/3$ | 39.40 | 37.19 | 40.69 | 38.37 | 0.35 | 0.27 |
| 1000 | 20 | 1.1 | 1.1 | $N^3$ | 46.37 | 43.21 | 48.15 | 44.83 | 0.45 | 0.43 |
| 1000 | 20 | 1.1 | 1.1 | $N^3/3$ | 51.86 | 49.13 | 53.96 | 51.34 | 0.85 | 1.02 |
| 1000 | 20 | 1.1 | 1.5 | $N^3$ | 55.04 | 52.73 | 57.32 | 55.60 | 0.82 | 1.10 |
| 1000 | 20 | 1.1 | 1.5 | $N^3/3$ | 55.50 | 53.53 | 58.36 | 56.74 | 1.12 | 1.46 |
| 1000 | 20 | 1.5 | 1.1 | $N^3$ | 34.19 | 33.20 | 35.27 | 34.03 | 0.43 | 0.32 |
| 1000 | 20 | 1.5 | 1.1 | $N^3/3$ | 34.21 | 33.18 | 35.30 | 34.03 | 0.36 | 0.26 |
| 1000 | 20 | 1.5 | 1.5 | $N^3$ | 39.76 | 37.69 | 41.01 | 38.58 | 0.29 | 0.18 |
| 1000 | 20 | 1.5 | 1.5 | $N^3/3$ | 47.08 | 43.48 | 49.09 | 45.51 | 0.63 | 0.72 |

if the starting point is selected as $F$, then the total assembly cost becomes 2.9 s. Hence, new improvements for assembling group 1 components can be obtained by a different preference of the starting point.

## VI. CONCLUSION

In this paper, we explored the assembly time minimization problem for the chip mounter machines. This problem and its subproblems, namely the placement sequencing and the feeder configuration problems, were stated and formulated mathematically. Placement sequencing problem was formulated using nonlinear integer programming model and it was shown that it was actually a new generalization of the TSP.

After giving the formulation of the assembly time minimization problem, we proposed three heuristics for improving the former studies on chip mounter machines: 1) RRTLEM; 2) AFPP; and 3) EPD. RRTLEM is a local search heuristic based on LEM. On the other hand, AFPP was a fast, simple, and effective heuristic that took advantage of opportunities arising from the operations of the machine. EPD broke the grouping strategy of points with same turret time values. Former studies were improved and very close results to LB were obtained using these heuristics. Between the proposed heuristics, AFPP attracted the most attention because of its simplicity, effectiveness and low run time. RRTLEM and EPD outperformed AFPP but had a larger run time. In addition, the effectiveness of the proposed heuristics was shown by comparing with the performance of SA metaheuristic.

Another type of placement machine known as chip shooters shared many similar characteristics with chip mounters and their placement sequencing problem definitions were almost the same. Therefore, the gained experience and the developed techniques can be applied to them in a possible future work.

## APPENDIX

To reveal the best set of parameter values for SA, we conduct an experiment in the following manner. Recall that the set of parameter values for the parameters are as follows: $T : \{100, 1000\}$, $P : \{5, 20\}$, $a : \{1.1, 1.5\}$, $b : \{1.1, 1.5\}$, and nox: $\{1, 3\}$. This corresponds two 32 different set of parameter values.

We run each parameter value set 50 times for a PCB and recorded the minimum, maximum, average, and variance of the results. Table V shows the average of these results for ten homogeneous and ten structured random PCBs. The best results are marked in bold. Results indicate that the parameters in the first row perform well. The variance column indicates that it is also much more robust than others.

## Acknowledgment

The authors would like to thank the anonymous referees and the associate editor for a thorough and thoughtful review of a previous version of this manuscript. This final version is improved because of their editorial efforts.

## References

[1] M. Ayob and G. Kendall, "A survey of surface mount device placement machine optimisation: Machine classification," *Eur. J. Oper. Res.*, vol. 186, pp. 893–914, Apr. 2007.

[2] W. Wang, P. C. Nelson, and T. M. Tirpak, "Optimization of high-speed multistation SMT placement machines using evolutionary algorithms," *IEEE Trans. Electron Packag. Manuf.*, vol. 22, no. 2, pp. 137–146, Apr. 1999.

[3] Y. Crama, J. V. D. Klundert, and F. C. R. Spieksma, "Production planning problems in printed circuit board assembly," *Discrete Appl. Math.*, vol. 123, nos. 1–3, pp. 339–361, 2002.

[4] P. Ji and Y. F. Wan, "Planning for printed circuit board assembly: The state-of-the-art review," *Int. J. Comput. Appl. Technol.*, vol. 14, nos. 4–6, pp. 136–144, 2001.

[5] P. Csaszar, T. M. Tirpak, and P. C. Nelson, "Optimization of a high-speed placement machine using tabu search algorithms," *Ann. Oper. Res.*, vol. 96, nos. 1–4, pp. 125–147, 2000.

[6] T. M. Tirpak, "Design to manufacturing information management for electronics assembly," *Int. J. Flex Manuf. Syst.*, vol. 12, nos. 2–3, pp. 189–205, 2000.

[7] E. Duman, "Optimization issues in automated assembly of printed circuit boards," Ph.D. dissertation, Ind. Eng. Dept., Bogazici Univ., Istanbul, Turkey, 1998.

[8] J. Smed, M. Johnsson, and O. Nevalainen, "A hierarchical classification scheme for electronics assembly problems," in *Proc. TOOLMET Symp., Tool Environ. Develop. Methods Intell. Syst.* 2000, pp. 116–119.

[9] E. Duman and I. Or, "Precedence constrained TSP arising in printed circuit board assembly," *Int. J. Prod. Res.*, vol. 42, no. 1, pp. 67–78, 2004.

[10] J. C. Ammons, M. Carlyle, G. W. Depuy, K. P. Ellis, L. F. McGinnis, C. A. Tovey, and H. Xu, "Computer-aided process planning in printed circuit card assembly," *IEEE. Trans. Compon., Hybrids, Manuf. Technol.*, vol. 16, no. 4, pp. 370–376, Jun. 1993.

[11] E. Duman, "Modelling the operations of a component placement machine with rotational turret and stationary component magazine," *J. Oper. Res. Soc.*, vol. 58, pp. 317–325, Mar. 2007.

[12] A. F. Alkaya, E. Duman, and M. A. Eyler, "Assembly time minimization for an electronic component placement machine," *WSEAS Trans. Comput.*, vol. 7, no. 4, pp. 326–340, 2008.

[13] A. F. Alkaya and E. Duman, "A new generalization of the traveling salesman problem," *Appl. Comput. Math.*, vol. 9, no. 2, pp. 162–175, 2010.

[14] C. D. J. Waters, "A solution procedure for the vehicle-scheduling problem based on iterative route improvement," *J. Oper. Res. Soc.*, vol. 38, no. 9, pp. 833–839, 1987.

[15] G. Croes, "A method for solving traveling-salesman problems," *Oper. Res.*, vol. 6, no. 6, pp. 791–812, 1958.

[16] S. Lin and B. W. Kernighan, "An effective heuristic algorithms for the traveling salesman problem," *Oper. Res.*, vol. 21, no. 2, pp. 498–516, 1973.

[17] J. L. Bentley, "Fast algorithms for geometric traveling salesman problems," *ORSA J. Comput.*, vol. 4, no. 4, pp. 387–411, 1992.

[18] G. Gutin and A. P. Punnen, *The Traveling Salesman Problem and its Variations* (Combinatorial Optimization), vol. 12. Norwell, MA, USA: Kluwer, 2002.

[19] G. Dueck, "New optimization heuristics: The great deluge algorithm and the record-to-record travel," *J. Comput. Phys.*, vol. 104, no. 1, pp. 86–92, 1993.

[20] F. Li, B. Golden, and E. Wasil, "The open vehicle routing problem: Algorithms, large-scale test problems, and computational results," *Comput. Oper. Res.*, vol. 34, no. 10, pp. 2918–2930, 2007.

[21] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, 2003.

[22] E. Duman and I. Or, "The quadratic assignment problem in the context of the printed circuit board assembly process," *Comput. Oper. Res.*, vol. 34, no. 1, pp. 163–179, 2007.

**Ali Fuat Alkaya** was born in Hatay, Turkey, in 1976. He received the B.S. degree in mathematics from Koc University, Istanbul, Turkey, and the M.S. degree in computer engineering and Ph.D. degree in engineering management from Marmara University.

He is a Faculty Member with the Computer Engineering Department, Marmara University. His current research interests include scheduling, analysis of algorithms, meta-heuristics, and combinatorial optimization.

**Ekrem Duman** was born in Afyon, Turkey, in 1967. He received the B.S. degree in electrical and electronics engineering and the M.S. and Ph.D. degrees in industrial engineering from the Bogazici University, Istanbul, Turkey.

He was a Faculty Member with the Industrial Engineering Department, Ozyegin University, Istanbul. His current research interests include industrial applications of operations research, scheduling, and data mining.