Discrete Applied Mathematics 192 (2015) 2–16

Contents lists available at ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Combining and solving sequence dependent traveling salesman and quadratic assignment problems in PCB assembly☆

Ali Fuat Alkaya[a,*], Ekrem Duman[b]

[a] Marmara University, Faculty of Engineering, Computer Science and Engineering Department, Goztepe, Istanbul, Turkey
[b] Ozyegin University, Faculty of Engineering, Industrial Engineering Department, Cekmekoy, Istanbul, Turkey

ARTICLE INFO

Article history:
Received 10 March 2013
Received in revised form 28 February 2015
Accepted 16 March 2015
Available online 9 April 2015

Keywords:
PCB assembly
Sequence dependent TSP
Metaheuristics
Artificial bee colony
Simulated annealing

ABSTRACT

In this study we undertake the optimization of chip shooter component placement machines which became popular in assembling printed circuit boards (PCB) in recent years. A PCB is usually a rectangular plastic board on which the electrical circuit to be used in a particular electronic equipment is printed. The overall optimization of the chip shooter placement machines leads to a very complicated optimization problem which we formulate here for the first time (without any simplifying assumptions). However, it is possible to decompose the problem into placement sequencing problem and feeder configuration problem which turn out to be sequence dependent traveling salesman problem (SDTSP) and Quadratic Assignment Problem (QAP), respectively. We use simulated annealing metaheuristic approach and the heuristics developed for the SDTSP in an earlier study to solve these two problems in an iterative manner. We also attempt to solve the combined overall optimization problem by simulated annealing and artificial bee colony metaheuristics and compare their performances with the iterative approach. The results are in favor of iterative approach.

1. Introduction

In this study, we tackle one of the most difficult combinatorial optimization problems arising from the use of component placement machines that are used to populate electronic components on printed circuit boards (PCB). These machines brought speed and reliability to the component placement operations however, no matter how fast they are, their operations are needed to be optimized if their capacities are wanted to be utilized to the fullest extent. Therefore, optimization issues regarding the automated assembly of printed circuit boards attracted the interest of researchers for several decades.

A PCB is usually a rectangular plastic board on which the electrical circuit to be used in a particular electronic equipment is printed and the locations of the electronic components to be mounted are identified. PCBs are used extensively in a variety of products such as: computers, calculators, robots, remote controllers, business telephones, cellular phones, and many electronic instruments. In fact, the PCB market revenue is expected to have a compound annual growth rate of 7.3% through 2016 [40].

☆ This work was supported in part by the Scientific and Technological Research Council of Turkey (TUBITAK) under the project ID 108M198. Work of A.F. Alkaya was supported by Marmara University Scientific Research Committee under the project ID FEN-A-150513-0172.

* Corresponding author.
E-mail addresses: falkaya@marmara.edu.tr (A.F. Alkaya), ekrem.duman@ozyegin.edu.tr (E. Duman).

http://dx.doi.org/10.1016/j.dam.2015.03.009
0166-218X/

The PCB production process is an assembly line that involves solder paste, component placement, and solder reflow operations. A placement 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 [13,42]. Nowadays, there are many types of placement machines available, such as sequential pick-and-place, multi-head, dual-delivery, turret type, multi-station, concurrent pick-and-place, etc. [29,30,8,7,26]. Various types of placement machines have different characteristics and restrictions [43,33,44]. Thus, the PCB production scheduling process is highly influenced by the type of SMT placement machine being used. For a complete survey on PCB assembly, see [12,31].

Basically, the operations of these machines yield four major problems [16]. These are allocation of component types to machines, determination of board production sequence, allocation of component types to feeder cells (also called the feeder configuration problem) and determination of component placement sequence. In many other studies, this list is extended or shortened but the last two have great influence and hence importance in optimizing the PCB placement machines [12,41]. All of these problems are interdependent, that is solution of one affects the other. Depending on the principles of the machine, some may be trivially solved while in most cases they yield NP-Complete problems. Hence a solution aiming to achieve the optimum in all problems simultaneously is very difficult to find, if not impossible.

Generally, the determination of component placement sequence is akin to traveling salesman problem (TSP) or its variants such as the Precedence Constrained TSP, the multiple TSP or vehicle routing problem [19,18,25]. On the other hand, feeder configuration problem turns out to be a Quadratic Assignment Problem (QAP) in machines having movable feeder magazine [20].

In this study, we focus on optimizing the operations of chip shooter placement machines because they are widely accepted as the latest technology high speed placement machines [10,27,28,39,6]. Basically, these machines have multiple rotating pick and place heads and a linear feeder carriage that moves horizontally and makes available one of the next components to be placed in line with the placement head. At the same time, the placement head makes a placement over the PCB once the placement location is adjusted below it by the two dimensional movements of the PCB carrier. The operations of these machines yield two major and interdependent problems: the placement sequencing problem and the feeder configuration problem. Given the feeder configuration, the placement sequencing problem actually turns out to be a newly introduced Sequence Dependent TSP (SDTSP) [4] while given the placement sequence, the feeder configuration problem turns out to be a QAP.

The objectives in the two problems are not exactly parallel. For the SDTSP the objective is to minimize the total assembly time needed to complete the assembly of a board. In this regard, it is parallel with the objective of the planner and thus it is a better representative of the real problem faced. However, in QAP, the objective is to minimize the total movement of the feeder carriage which might help in decreasing the total assembly time in return.

For the optimization of chip shooter machines, in the literature there are a few studies which solve the placement sequencing and the feeder configuration problems in an iterative manner [23,35]. In those studies however, a simplified version of placement sequencing problem is assumed which results in a standard TSP. On the other hand, the problems can be solved simultaneously using meta-heuristics. In our study we use both approaches and compare their performances with a SDTSP setting for the placement sequencing problem.

Specifically, we implement the simulated annealing (SA), and the artificial bee colony (ABC) meta-heuristic approaches and use them both for the solution of the combined problem and the solution of the QAP in the iterative approach. The reason for choosing SA is that it performed pretty well for feeder configuration problem in an earlier study [20]. On the other hand, the reason for choosing ABC is that it is relatively a new approach and its performance is not explored yet for combinatorial optimization problems [32].

In this paper, our contribution is providing a nonlinear integer programming formulation of the combined problem of placement sequencing problem and feeder configuration problems arising from the operation principles of chip shooters. In these formulations every detail is considered and no simplifying assumptions are made. We use simulated annealing metaheuristic approach and the heuristics developed for the SDTSP in an earlier study to solve these two problems in an iterative manner. We also attempt to solve the combined overall optimization problem by simulated annealing and artificial bee colony metaheuristics and compare their performances with the iterative approach. The results are in favor of iterative approach. It outperforms the metaheuristic approach by up to 18% where even a 1% improvement in assembly time has a significant impact for the manufacturer.

In the next section the operation principles of chip shooter machines and problems arising from them are given in detail. In section three, we give the notation used for the formulations and then the formulations of the problems. In section four, solution methodologies proposed for these problems are given. Section five includes the discussion on the results obtained on both real life PCB assembly problems and on PCB problems generated synthetically. Section six concludes by providing a summary of the study and directions for further study.

2. Chip shooter placement machines

There are various models of chip shooters with different operation parameters. However, in basic terms, they consist of three parts (see Fig. 1):

- A board carrier: The PCB lies on this carrier which is able to move horizontally and vertically in a concurrent manner. To achieve this concurrency, the carrier is controlled by two independent motors.

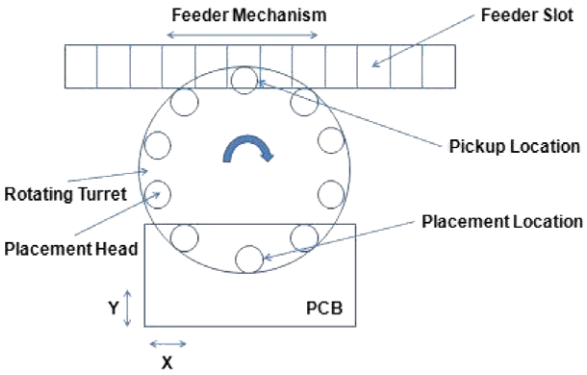

Fig. 1. Chip shooter machine.

- A feeder carriage (rack): The feeder carriage consists of the so called slots. It is linearly (horizontally) movable. A feeder is a reel that stores components of a single type, and is attached to a slot on the feeder carriage.
- A rotating turret (carousel): This is a device which is able to transport components from the feeder carriage to the board by rotating clockwise. The turret contains an even number of placement heads (H) that are arranged on the perimeter of the turret.

In the following subsections, firstly, the operation principles and problems emerging from the operations of chip shooter machines will be given in detail. Then, the impacts of varying turret rotation time of these machines and the assumptions used in the literature to cope with the particular machine specifications are summarized briefly.

The placement operation is performed by the placement heads that rotate between a fixed pickup and fixed placement location. Each assembly head has several nozzles to pick up components of various sizes. A large nozzle is used to pick up and place large components. The function of the feeder carriage is to move in the x-direction to align the correct component feeder under the pickup head in the fixed pickup location. The board carrier secures the PCB and positions it under the placement head in the fixed placement location where the loaded component is mounted. The movements of these three parts are almost independent of one other (the slight interdependency will be explained later). Since individual activities of a chip shooter machine all happen simultaneously, the resulting model of its movement is very complex. But by itemizing the movements of the machine, we can extract the modus operandi of the placement machine.

2.1. Operation principles

The placement operation is performed by the placement heads that rotate between a fixed pickup and fixed placement location. Each assembly head has several nozzles to pick up components of various sizes. A large nozzle is used to pick up and place large components.

The function of the feeder carriage is to move in the x-direction to align the correct component feeder under the pickup head in the fixed pickup location.

The board carrier secures the PCB and positions it under the placement head in the fixed placement location where the loaded component is mounted.

The movements of these three parts are almost independent of one other (the slight interdependency will be explained later). Since individual activities of a chip shooter machine all happen simultaneously, the resulting model of its movement is very complex. But by itemizing the movements of the machine, we can extract the modus operandi of the placement machine.

a. turret rotates (indexes) one step (i.e. rotates 360/H degrees)
b. board carrier aligns the new placement point under the placement head
c. placement heads rotate if necessary to align the suction nozzle carrying the component to be placed
d. the feeder carriage moves to place the correct component feeder under the pickup head
e. the placement head makes the placement
f. the pickup head retrieves the component.

Operations a–d are performed concurrently which is followed by the concurrent movement of items e–f. Hence, there is sequential order between these two groups of operations. These movements are repeated for each component of the PCB.

The time for retrieval and placement (operations e and f) is normally assumed to be equal and the same for all components and no optimization of this time is possible [10]. Therefore, after multiplying it with number of components (N), it is added as a constant to the assembly time formulations.

On the other hand, as there are three moving parts in a high speed chip shooter (board carrier, feeder carriage and carousel), each moving part has to wait for other two parts to complete their movements before the next component can

be picked up or placed. Hence, the mechanism that takes the longest to complete the operation dictates the placement rate during each pickup and placement. The placement time for any component is the maximum of three movement times (turret rotation time, feeder carriage movement time and board carrier movement time) plus the constant time for retrieval and placement. The operation time for aligning the suction nuzzle carrying the component to be placed is mostly very smaller than the indexing of the turret, hence does not contribute to the cost calculation.

The reader should easily see that the component gripping sequence is $H/2$ indices/heads ahead of the component placement sequence. Also, it can be observed that the actual benefit of chip shooters is its high speed because the pickup and placement operations are performed simultaneously.

This machine type is usually called a chip shooter machine but some researchers choose to use the term concurrent chip placement machine [45,37,5].

Examples of common chip shooter machines include the Fuji CP4, CP4-2, and CP4-3 machines, which have 12 mounting heads and 160 feeder slots, and the Fuji CP6 machine, which has 20 placement heads and 160 feeder slots [22]. In Fig. 1, we illustrate a chip shooter machine with 10 placement heads and 12 feeder slots. Chyu and Chang [10] talk about CP 642 and CP 742, which are more recent chip shooter models. Another study is on CM82 chip shooter machine with 18 placement heads and 100 feeder slots [38].

2.2. Problems emerging from the operations of chip shooter machines

Given a PCB type, calculation of assembly time is determined by the placement sequence of components and assignment of component types to feeder slots. Hence, there are at least two problems that await optimization in order to obtain a smaller assembly time. Once a feeder configuration is given, determination of component placement sequence (placement sequencing problem) recalls the well-known Traveling Salesman Problem (TSP) since a tour is developed for the components to be placed [37,5]. In fact it turns out to be a variant of TSP (the SDTSP) where the reasons of why will be made clear in the following section. On the other hand, once a placement sequence is given, determination of assignments to slots (feeder configuration) is similar to a Quadratic Assignment Problem (QAP), since the feeders are assigned to slots on the feeder carriage and the cost of the assignment is affected by the location of the other feeders [37,5]. For chip shooter machines, the placement sequencing and feeder configuration problems are highly interdependent. Feeder configuration influences the placement sequence since the time spent between two consecutive placements depends on the feeding time (thus, the feeder configuration) of the next component to be placed to the head. On the other hand, placement sequence influences the feeder configuration, since the amount (time) of linear feeder carriage movement depends on the distance between the feeder locations of the consecutively placed components (placement sequence). Thus, the inherent operation principles of chip shooters require us to solve placement sequencing and feeder configuration problems concurrently. Let us call this combined problem as the master problem. Both the SDTSP and the QAP are NP-Complete problems for which general algorithms for solving all instances are not available. Thus, optimizing feeder configuration and placement sequencing problems simultaneously becomes a more challenging problem.

2.3. Turret rotation time

A property of chip shooter machines makes the placement sequencing problem much more difficult than a standard TSP. This is the varying speed of the turret according to the transferred components. This property of the machines is mostly overlooked by researchers.

The rotation of the turret is performed at high speed. Hence, substantial centrifugal forces work on the components transferred. These forces cause the component to be dislocated or even to be lost. Their strength depends on various characteristics of the components such as size, weight and type of packaging. Thus the control unit of chip shooters determines the possible rotation speed depending on the individual component type. For this purpose, heavier component types are assigned to rotation speed classes which allow only a comparatively slow transfer. The feasible rotation speed does not only depend on the component that is being placed next or the component that has just been picked up but on the entire set of components that is transferred in the turret at that moment [24]. When any of the heads picks up a heavier component type, the rotation speed of the turret is reduced for $H/2$ steps until it is discharged at the placement position. This phenomenon complicates the problem formulations because while determining placement sequence of components this complication must also be taken into consideration. Thus, the placement sequencing problem of chip shooter machines turns out to be a newly introduced generalization of the TSP: the Sequence Dependent TSP. How the placement sequencing problem of a placement machine with similar characteristics is formulated as an instance of SDTSP is given in [4]. In the rest of this paper, turret time for weight category k is represented by τ_k.

There are many studies performed for optimizing the operations of chip shooter machines. A survey of these studies is given in [3]. However, there are very few studies taking the varying turret rotation speed into consideration [10,22,24]. To our best, this is the first study that takes into consideration the varying turret rotation speed while formulating the operation principles. Crama et al. [11] only remind the reader about the change of turret rotation speed depending on the carried components but they neglect this property while modeling the machine in order to obtain simplicity and by arguing that this property of the machine will have only marginal effects to the assembly times, which we believe is a wrong argument as explained above. Assumptions for machine specifications and turret time values used in literature are given in detail in [3].

3. Problem formulations

In this section, we give nonlinear integer programming formulation of the placement sequencing, feeder configuration and the master problem which considers solving the former two problems concurrently. The following formulations assume that the placement machine has different turret time values for different component types which is the real case as we pointed out above.

3.1. Notation

Below, the notation used for modeling the operations of chip shooter machines is given.

H: number of heads on the turret,
N: total number of components to be placed,
n: number of component types, number of feeder slots (one component type is assigned only one slot)
K: number of weight categories
N_k: number of components in each weight category k, $k = 1, 2, \ldots, K$
n_k: number of component types in each weight category k, $k = 1, 2, \ldots, K$
d: number of heads between pickup and placement heads ($d = H/2 - 1$)
c_{it}: component type matrix ($N \times n$) indicating the component type for each component

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

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

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

τ_k: turret rotation time for each weight category k, (when $k = 1$, the turret rotation speed is the maximum, that is, turret rotation time is minimum) (τ is used if weight differences among components is disregarded)
$CF1(x, y)$: time required for the board carrier to move from component x to component y.
$CF2(r, s)$: time required for the feeder carriage to travel from feeder r to feeder s.
p: is the placement order of a component's placement on the board.
In order to express the placement sequence we need to define decision variable x_{ip} which denote the placement of component i in placement order p.

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

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

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

On the other hand, in order 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.

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

In this notation, the definition of the cost function $CF1(x, y)$ is as follows:

$$CF1(x, y) = \frac{d(x, y)}{v} \tag{1}$$

where v is a predefined speed value and $d(x, y)$ is the distance between points x and y. Note that due to simultaneous and independent movements of board carrier in the x–y plane, $d(x, y)$ is defined as $\max |x_1 - y_1|, |x_2 - y_2|$ which turns out to be Chebyshev metric, where x_1 and x_2 represent the x and y-coordinates of point x, respectively. On the other hand, $CF2(r, s)$ gives time required for the feeder carriage to travel from feeder r to feeder s. We assume a linear speed for the feeder carriage (no acceleration or deceleration) and it can be stated by the following equation.

$$CF2(r, s) = |r - s| f \tag{2}$$

where f is a predefined time value for the feeder magazine to move one slot. As it is pointed out before, the operations of chip shooter machines result in two interdependent optimization problems. These are the placement sequencing and the feeder configuration problems. Formulation of both of these problems and the combined problem which we call as the master problem here will be given in the following subsections.

3.2. Master problem

Recall that the ultimate objective of the optimization problem for chip shooter machines is to minimize the total assembly time. That is, the objective is to find a placement sequence of components and a feeder configuration so that the assembly

process is completed in the shortest time possible. So the complete and integrated model can be formulated by firstly defining the cost function of assembling component j just after component i in pth position (C_{ijp}).

$$C_{ijp} = w_{ijp} \max \left\{ \begin{array}{l} CF1(i,j), \\ \max\limits_{m=0}^{d} \left\{ \sum\limits_{l=1}^{N} \sum\limits_{t=1}^{n} \sum\limits_{k=1}^{K} \tau_k g_{tk} c_{lt} x_{l,p+m} \right\}, \\ \sum\limits_{t_1=1}^{n} \sum\limits_{t_2=1}^{n} \sum\limits_{r=1}^{n} \sum\limits_{s=1}^{n} \sum\limits_{u=1}^{N} \sum\limits_{v=1}^{N} CF2(r,s) x_{u,p+d} x_{v,p+d+1} y_{t_1,r} c_{u,t_1} y_{t_2,s} c_{v,t_2} \end{array} \right\}. \tag{3}$$

In this formulation, first term in 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 the turret time values of the following $d+1$ components (including component j) that are carried by the heads. This is stated by a maximum function and within the maximum function the turret time values for the corresponding components to be placed in pth, $(p+1)$st, ..., $(p+d)$th placement orders are calculated. While the turret is rotating one step, the feeder magazine must move to get ready for the pickup of the component to be placed in the $(p+d+1)$th position. Hence, the third term finds the time for the feeder magazine to move from slot r to slot s where the r and s are the slots of the $(p+d)$th and $(p+d+1)$th components, respectively. This is achieved by incorporating the decision variables x_{ij} and y_{tr}.

It is important to note a detail about indices used in formulations. 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. So, traveling along the path restarts after a previous tour ends. Hence, in the formulations, if an index exceeds its upper limit, it should be considered as if the counting of indices 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.

After removing the nonlinearity in C_{ijp} by introducing D_{ijp} as the dominating factor in C_{ijp} definition, we obtain the following nonlinear integer programming formulation of the problem. Note that this formulation incorporates both x_{ip} and y_{tr} variables.

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

s.t.

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

$$D_{ijp} - w_{ijp} \sum_{l=1}^{N} \sum_{t=1}^{n} \sum_{k=1}^{K} \tau_k g_{tk} c_{lt} x_{l,p+m} \geq 0, \quad i,j,p = 1,\ldots,N,\ m = 0,\ldots,d, \tag{6}$$

$$D_{ijp} - w_{ijp} \sum_{t_1=1}^{n} \sum_{t_2=1}^{n} \sum_{r=1}^{n} \sum_{s=1}^{n} \sum_{u=1}^{N} \sum_{v=1}^{N} CF2(r,s) x_{u,p+d} x_{v,p+d+1} y_{t_1,r} c_{u,t_1} y_{t_2,s} c_{v,t_2} \geq 0, \quad i,j,p = 1,\ldots,N, \tag{7}$$

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

$$\sum_{i=1}^{N} x_{ip} = 1, \quad p = 1,\ldots,N, \tag{9}$$

$$\sum_{i=1}^{N} w_{ijp} = x_{jp}, \quad j,p = 1,\ldots,N, \tag{10}$$

$$\sum_{j=1}^{N} w_{ijp} = x_{i,p-1}, \quad i,p = 1,\ldots,N, \tag{11}$$

$$\sum_{r=1}^{n} y_{tr} = 1, \quad t = 1,\ldots,n, \tag{12}$$

$$\sum_{t=1}^{n} y_{tr} = 1, \quad r = 1,\ldots,n, \tag{13}$$

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

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

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

We introduce several constraint sets to the problem. Constraint set (5) imposes the requirement that placement time for component j (component j being placed after component i in pth position) cannot be smaller than the travel time from i to j. Constraint set (6) (representing $d + 1$ different constraint sets) impose that placement time for component j cannot be smaller than the turret time values associated with the incoming components for placement. Constraint (7) guarantees that placement time for component j cannot be smaller than the movement of feeder magazine. Constraint (8) guarantees that each component is assigned to a position and constraint (9) guarantees that each position is occupied by only one node. There is no need for constraints that are used to guarantee subtour elimination, because placing each node in exactly one position is achieved by these constraints. Constraint (10) states that if component j is assigned to pth position, then exactly one component precedes j in the travel sequence. Similarly (11) states that if component i is assigned to $(p - 1)$th position then exactly one component proceeds it in the travel sequence. Constraint (12) guarantees that each component type is assigned to a feeder slot and constraint (13) guarantees that each feeder slot is occupied by only one component type. Constraints (14)–(16) are the binary constraints for decision variables x_{ip}, w_{ijp} and y_{tr}.

3.3. Placement sequencing and feeder configuration problems

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. Below we define the way to do this.

Given a feeder configuration, the placement sequencing problem can be easily formulated by using the same objective function and same set of constraints excluding (12), (13) and (16). However, in the objective function and constraint (7) 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 objective function and set of constraints (5)–(7), (12), (13) and (16). In the objective function and constraints (5)–(7) we must use the values of x_{ij} variables given through the placement sequence.

4. Proposed solution methods

The goal of the master problem is to minimize the total assembly time of a chip shooter machine for a given PCB. Finding the exact solution is computationally very expensive for the master problem even for small instances because it requires us to solve two NP-Complete problems concurrently. Therefore, effective heuristic methods should be used for solving the master problem.

We designed two different solution approaches for the master problem. One way is using meta-heuristics for the combined representation of the problem. The other way is to use an iterative scheme between the sub-problems. We start with describing the iterative approach.

4.1. Iterative approach

The master problem is a very complicated problem as its mathematical model is presented. However, as described in the problem formulation section, it can be decomposed into two well-known and well studied subproblems. For this reason, once it is decomposed we can apply the solution methods that are proven to perform well on the subproblems and solve the master problem iteratively. We believe that this iterative method will provide high quality solutions.

In the iterative approach, for a given feeder configuration we try to minimize assembly time by means of searching for the best placement sequence. On the other hand, for a given placement sequence we try to minimize either total assembly cost or the total movement distance of feeder carriage in the search for best feeder configuration. However, we should note that, in the latter case, an overall optimal solution or even an improvement at each iteration cannot be guaranteed. In our design, the stopping criterion is a predefined iteration number. In the iterative approach, the methods to be used for obtaining solutions to the sub-problems are the ones that are proven to demonstrate good performances.

In the literature there are many solution approaches developed and applied for the QAP [14,1,36,9,21]. However, there are two recent studies which are focused on the QAP in the context of PCB assembly [20,34]. In both of these studies, several meta-heuristics are implemented and analyzed with different parameter settings and it is found that SA (simulated annealing) is the best performing meta-heuristic in the context of PCB assembly. Therefore, in this study the preferred solution technique applied for the QAP is SA.

On the other hand, SDTSP is a recently introduced TSP generalization and there are very few studies on it [4,2]. By combining the Assembly Time Minimization Algorithm (ATMA) proposed by Duman [17] for optimizing the operations of a similar placement machine, Alkaya [2] proposes several solution techniques for the SDTSP, but the most promising are Adjust First Point Procedure (AFPP), Postpone Deviant (PD) and Record-to-Record Travel with Local Exchange Moves

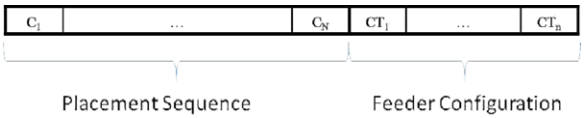

Fig. 2. Combined representation of a solution.

(RRTLEM). Furthermore, in that study, it is shown that those specifically developed heuristics outperform SA. Therefore the preferred solution techniques applied for the SDTSP is ATMA, AFPP, RRTLEM and PD which should be applied in an additive (i.e. one after the other) manner.

Even though it is stated that the placement sequencing problem is a hard one even to formulate for the chip shooters or machines with similar characteristics (i.e. having different turret time values for different weight categories), a solution methodology is developed in [17]. The developed methodology is based on the observation that mixing heavy and light components in the placement sequence is quite inefficient. So, 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 ttk 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 [17] for the solution of the placement sequencing problem. In ATMA, after finding the TSP routes for each weight category, the routes are connected in a cost-efficient way.

Specifically, among all group 2 components, closest one to the last component of tour of group 1 is chosen as the starting point for tour of group 2 and so tours for group 1 and group 2 is connected. This process is repeated for group 2 and group 3 components and then for group 3 and group 4 components.

RRTLEM is a hybrid of record-to-record travel (RRT) and local search moves [2]. RRT is, as a deterministic variant of SA, developed by Dueck and it is shown that the quality of the computational results obtained so far by RRT is better than SA [15]. 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*. RRTLEM is an improvement algorithm that combines RRT with local search moves. We call it RRT with local exchange moves (RRTLEM). It is a general framework that consists of simple iterative statements [2].

Another heuristic developed for the SDTSP is the Adjust First Point Procedure (AFPP) which is a simple and effective algorithm. The idea behind AFPP is very simple. Firstly, an initial tour is built by using ATMA. Then for each point i in route 1, the route is modified such that the tour starts from point i. Naturally, this modification results in a change of last point of route 1 which causes a need for a reconnection of route 2 to the modified route 1 possibly through a new connection point. So, route 2 is modified. Similarly route 3 and route 4 are modified. At the end, the point that minimizes cost of the tour is chosen as the first point for the assembly [2].

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. For example if a component is far away (i.e. deviant) from the other components in the same group, it may be wise to place that component when the machine is slower (i.e. when a heavier group of components are being placed). This is the idea behind PD where, lighter components are inserted among heavier components if cost effectiveness is the case [2].

4.2. Solving the master problem directly

Using meta-heuristics to solve the master problem is an alternative approach. For this, we decided to use the SA and the ABC meta-heuristics. SA is a good performing meta-heuristics on the QAP [20,34]. ABC is a comparably new meta-heuristic and is not fully explored on discrete problems yet [32]. Therefore, we believe that ABC will bring new perspectives to combinatorial optimization researchers.

A combined representation of a solution to the master problem consists of two parts: placement sequence and feeder configuration (Fig. 2). In the figure, number of components is denoted by N and number of component types is denoted by n.

Our SA implementation is the same as the one given in [20]. The parameters for this implementation are 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.
- 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 become 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 (R): Greater values of R correspond to slower cooling, that is, more exchanges occurring when there is a greater likelihood of inferior exchanges being accepted. In numerical computations, R is set to either 5 or 20.

- Increase ratio in iteration number at each setting (b): After a predetermined number of iterations, R is set to $R \times b$ (i.e. $R := R \times b$). In numerical computations, b is set to either 1.1 or 1.5.

ABC algorithm is inspired from foraging behavior of honey bees. In a bee colony, in order to maximize the amount of food collected, honey bees collaborate with each other. In that organization, there are three types of bees; employee bees, onlooker bees and scout bees. Employee bees are responsible for determining the quality of food source for onlooker bees which are waiting response from them. They dance for onlooker bees in order to exchange information which gives information on the quality of food source they have just visited. With the collective information from the employee bees, onlooker bees choose the food sources and try to find better ones. Scout bees are responsible for finding new food sources independently from the other bees. In this algorithm, all bees have the ability to keep their best solution.

The steps of the ABC are as follows;

i. Initialize the solutions randomly.
ii. Send employee bees to find solutions.
iii. From the probability calculations of employee bees, send onlooker bees and find new better solutions.
iv. Send the scout bees for abandoned solutions and generate new solutions.
v. Keep the best solution so far.
vi. Repeat ii–v until the stopping criteria are met.

In our implementation the stopping criterion is repeating the first four steps a number of times and it is one of the parameters of the algorithm called as number of cycles (*noc*). Apparently, a high value for the *noc* parameter gives the opportunity to look on broader solutions. Secondly, *limit* parameter determines the abandoned solutions and replaces them with new ones with scout bees. If the *limit* value is small, the improvement possibilities of a solution can be left unfinished and it is replaced with a new solution. On the other hand, an excessively high *limit* value causes the algorithm to wander about the same solutions' neighbors. Third parameter is the colony size (*cs*) which denotes the number of employee and onlooker bees in the colony. Number of scout bees is also equal to the number of onlooker bees.

The important point while applying the above mentioned meta-heuristics to the master problem is the decision on neighbor selection method. In our implementation, a neighbor solution is obtained by exploiting pairwise exchanges in the placement sequence and/or feeder configuration. Recall that a solution of the master problem is composed of feeder configuration and placement sequence sub-solutions. In order to choose a neighbor solution of the current solution, one can change the feeder configuration or the placement sequence iteratively based on a rule, or both of them. In this context, we developed five different neighbor choosing techniques:

1. To decide on the sub-solution to be changed: give equal chance to both of them.
2. To decide on the sub-solution to be changed: give weighted chance proportional with the square of the number of components and number of component types.
3. Make changes on the sub-solutions one-by-one (one after another).
4. Make changes on both the placement sequence sub-solution and feeder configuration sub-solution simultaneously.
5. Make changes on the placement sequence and then on the feeder configuration a number of times equal to the square of number of components and square of number of component types, respectively.

For each meta-heuristic, apart from their inherent parameters, the above neighbor choosing technique (*nct*) is an additional parameter.

In the next section, results of computational studies are given together with their discussions.

5. Results and discussion

After designing the algorithms, extensive computational tests are run to reveal their performance. In these computational tests, the machine specifications used by Ellis et al. are used with some rounding and linearization [22,3]. Specifically, the board carrier has a speed of 280 mm per s, four turret time values for the four different weight categories are: 0.15, 0.19, 0.24 and 0.29 s whereas the feeder mechanism moves first slot in 0.18 s and each additional slot counts for 0.045 s.

On the other hand, we used two sets of PCB data in the tests. First set consists of eight real PCB data which are also used in [20]. Number of components (N) and number of component types (n) are given in Table 1.

Second set consists of randomly generated PCB data. The total number of components to be placed on a board, N, is set to 100 and number of component types, n, is set to 52. The number of component types in groups 2, 3 and 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 rest of the components were in group 1 that is composed of 40 component types and the placement number of each one is probabilistically equal. 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 at the same coordinate. There are two types of random PCBs; homogeneous boards and 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. A research on real PCB manufacturing systems reveal the fact that the components in the heavier groups appeared to be placed more closely. Thus, structured board types more or less represent the real PCBs. But

Table 1
Properties of real PCBs.

Board	Code	n	N
B1	PS11AK08-9	52	146
B2	PS11AK1011	48	159
B3	PS11AK12-7	54	152
B4	PS11AK15-4	40	170
B5	PS11AK16-3	47	261
B6	PS11AK16-4	49	262
B7	PS11AK16-5	50	280
B8	PS11AK17N3	58	185

for the sake of a complete study, we will investigate the performance of our algorithm in both PCB types. The data generator generated 100 instances for both PCB types and they are kept as input files. In this section, any value regarding the random PCBs appearing in a comparison table is the average of 100 PCBs, unless stated otherwise.

The basic and most important comparison criterion in PCB placement machines is the total assembly time of a given PCB. A second criterion for comparing the heuristics is the run time. Run time of a heuristic is important because a heuristic that computes best results in irrational amount of time is of no value. Thirdly, we provide the standard deviation of the total assembly time values of the applicable experiments. They are given in parentheses.

To demonstrate the complexity of the problem, we try to obtain the exact solutions of reasonably sized instances. The above-given master 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 cubically with the number of components and the number of equations scale cubically with the number of component types and number of heads. For example, for a board with 30 components, seven component types to be assembled on a real chip shooter with six heads between pickup and placement heads, number of variables and number of equations are 27 949 and a value >1.62 million, respectively. For a moderate PCB with 100 components and 50 component types, the above numbers increase to >1 million and a value >60 million. Therefore, obtaining the exact result in a reasonable time seems to be impossible with the current computing technologies.

5.1. Using iterative method for solving the master problem

In the iterative approach, the methods to be used for obtaining solutions to the sub-problems are the ones that are proven to demonstrate good performance. Specifically, for the placement sequencing problem ATMA, AFPP, RRTLEM and PD are applied additively. For the feeder configuration problem (as an instance of QAP) simulated annealing meta-heuristic is applied. The best performing parameters for the QAP are identified as $T = 1000$, $R = 20$ $a = 1.5$ and $b = 1.1$ [20].

In the iterative approach, firstly given a random feeder configuration, SDTSP is solved by ATMA, AFPP, RRTLEM and PD. Then, given the placement sequence (as a solution of SDTSP), SA starts to find a solution to the feeder configuration problem considering it as an instance of QAP. Then, once more SDTSP is solved with the updated feeder configuration and this iteration is repeated 20 times.

During the utilization of SA there are two strategies that SA may use: calculating the total cost according to actual feeder speed values and thus making decisions on total assembly cost, or making calculations regarding the problem solely as an instance of QAP and taking the feeder speed as one cell per second. In the first strategy we try to minimize the total assembly cost where the objective function is the one given in (3) and in the second one we try to minimize the total number of steps taken by the feeder carriage where the objective function is the one of classical QAP formulation.

Another strategy to be revealed out for SA is the starting feeder configuration. In each iteration SA may use the feeder configuration found in the previous iteration as the starting feeder configuration or may start from a new random feeder configuration. Using the former best feeder configuration may cause the algorithm to wander around the same solution's neighbors whereas starting with a new one may cause the algorithm forget its experience. Combining these strategies makes four different setups. They are defined and named in Table 2.

An example of the results of the iterative 1 approach on a real PCB is given in Table 3. In these results, it is interesting to observe that absolute improvement in each iteration is not the case. See that, for example, in the 11 th iteration assembly cost after solving SDTSP is 63.78. However, in the next iteration SA tries to minimize the total movement of the feeder carriage using the current placement sequence and the cost of QAP decreases from 410 to 396. However, this QAP solution increases the total assembly time to 63.94. That's why the assembly cost shows a random behavior.

5.2. Using meta-heuristics for solving the master problem

For the meta-heuristics implemented in this study, we followed the following work plan to find out their best performing parameters. Firstly, each parameter set for a heuristic is run 10 times on the real PCB data and average and minimum of these 10 runs are recorded. Their results are analyzed and the best performing parameters are noted. After that, the best performing parameter sets are used on the random PCB sets.

Table 2
Definitions of four different iterative cases.

Name	Definition
Iterative 1	SA tries to improve the solution based on QAP cost In each iteration SA starts from the best solution found in the previous iteration
Iterative 2	SA tries to improve the solution based on total assembly cost In each iteration SA starts from the best solution found in the previous iteration
Iterative 3	SA tries to improve the solution based on total assembly cost In each iteration SA start from a new solution
Iterative 4	SA tries to improve the solution based on QAP cost In each iteration SA start from a new solution

Table 3
Results of iterative 1 method on an example PCB (B6).

Iteration	Assembly cost after solving QAP	Cost of QAP	Assembly cost after solving SDTSP
0	0	438	66.5
1	65.91	438	64.03
2	64.03	426	64.03
3	63.84	406	64.36
4	64.36	416	64.36
5	64.36	416	64.36
6	64.36	416	64.36
7	64.36	416	64.36
8	64.36	416	64.36
9	64.36	416	64.36
10	64.64	400	**63.78**
11	63.78	410	63.78
12	63.94	396	64.14
13	64.14	432	64.14
14	64.14	432	64.14
15	64.14	432	64.14
16	64.14	432	64.14
17	64.14	432	64.14
18	64.14	432	64.14
19	64.14	432	64.14
20	64.14	432	64.14

Table 4
Performance analysis for parameters of SA.

Par.	Val.	B1	B2	B3	B4	B5	B6	B7	B8	Avg.
T	100	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	**0.00**%
	1000	8.50%	4.60%	10.50%	0.70%	1.50%	2.80%	5.30%	5.20%	4.90%
R	5	2.70%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	**0.30**%
	20	0.00%	2.90%	1.60%	0.70%	1.00%	2.80%	5.30%	2.10%	2.00%
a	1.1	2.70%	0.00%	0.00%	5.40%	0.00%	0.00%	0.00%	0.00%	1.01%
	1.5	0.00%	1.10%	1.60%	0.00%	0.50%	0.60%	2.00%	2.10%	**0.98**%
b	1.1	2.70%	0.00%	0.00%	0.70%	0.00%	0.00%	0.00%	0.00%	**0.40**%
	1.5	0.00%	1.10%	1.60%	0.00%	0.50%	0.60%	2.00%	2.10%	1.00%
nct	1	0.60%	3.10%	2.90%	0.70%	1.20%	2.00%	4.40%	3.00%	2.20%
	2	0.00%	0.00%	2.60%	1.30%	0.00%	0.00%	0.00%	0.00%	**0.50**%
	3	1.10%	3.40%	0.00%	2.70%	1.50%	2.60%	2.80%	2.80%	2.10%
	4	20.70%	17.80%	20.50%	17.50%	20.60%	25.30%	24.20%	21.40%	21.00%
	5	5.20%	3.20%	3.70%	0.00%	1.30%	3.00%	3.60%	4.00%	3.00%

To find the best parameter value for a parameter of meta-heuristic, say T of SA, the parameter is fixed to value, say 100, and all results taken with various combinations of R, a, b, nox and nct are averaged to find average results for that value of the parameter. In this way, the average performance of the parameter value on all circumstances is presented. This analysis is repeated on all real PCB data.

SA has four parameters each having two different values. Additionally, the neighbor choosing technique (nct) has five values. These parameter values make a total of 80 different combinations. The performance analysis results for the parameters are given in Table 4 where the percentage deviations from the overall best solution are displayed.

Table 5
Performance analysis for parameters of ABC.

Par.	Val.	B1	B2	B3	B4	B5	B6	B7	B8	Avg.
noc	100	103%	95%	112%	77%	98%	106%	108%	122%	103%
	1 000	44%	42%	55%	32%	43%	54%	52%	55%	47%
	10 000	14%	11%	17%	6%	10%	13%	15%	20%	13%
limit	100	43%	37%	45%	28%	34%	39%	41%	53%	40%
	1 000	16%	13%	18%	6%	11%	17%	16%	23%	15%
	10 000	14%	11%	17%	7%	10%	13%	15%	20%	13%
cs	10	19%	14%	19%	7%	11%	14%	16%	21%	15%
	20	14%	13%	17%	6%	11%	13%	15%	22%	14%
	50	16%	11%	17%	7%	10%	16%	17%	20%	14%
nct	1	17%	12%	21%	8%	11%	17%	18%	23%	16%
	2	17%	12%	17%	6%	10%	14%	16%	23%	14%
	3	14%	12%	17%	7%	12%	15%	15%	21%	14%
	4	40%	32%	42%	24%	30%	38%	40%	47%	37%
	5	17%	11%	18%	7%	12%	13%	17%	20%	14%

Table 6
Results of all methods on real PCBs.

PCB	Assembly time (s)						Runtime (s)		
	SA	ABC	Iter.1	Iter.2	Iter.3	Iter.4	SA	ABC	Iter. (avg.)
B1	39.83 ± 2.08	43.53 ± 1.43	40.17	38.78	**38.10**	39.53	8	11	1269
B2	44.59 ± 1.81	46.58 ± 1.15	41.82	42.39	40.30	**39.99**	6	12	1790
B3	45.12 ± 2.22	46.87 ± 0.66	41.28	42.66	**40.96**	41.17	9	12	1184
B4	43.84 ± 1.83	45.13 ± 0.96	41.30	39.61	**38.85**	38.97	4	13	2440
B5	75.73 ± 2.10	83.26 ± 1.15	63.05	64.69	**60.32**	61.29	10	19	4196
B6	76.45 ± 1.75	85.24 ± 1.59	63.44	68.08	62.50	**62.22**	11	19	3780
B7	81.73 ± 2.76	89.96 ± 1.78	68.00	67.78	67.41	**67.04**	13	20	3714
B8	54.97 ± 2.55	60.37 ± 1.99	49.00	51.22	48.16	**46.98**	13	14	2827
Average	57.78 ± 2.14	62.62 ± 1.34	51.01	51.90	**49.58**	49.65	9	15	2650

We may reach the following results after analyzing Table 4 in detail. The parameter set $T = 100$, $R = 5$, $a = 1.5$ and $b = 1.1$ outperforms other alternative sets. From this set and the definitions of these parameters, we can infer that when the algorithm accepts inferior solutions for a long time, it has difficulty in bouncing back to better solutions.

When the *nct* parameter is analyzed we observe that method 2 gives the best result i.e. deciding on the sub-solution to be changed by giving weighted chance proportional with N^2 and n^2. In this way, for example, for a PCB with $n = 50$ and $N = 250$, we give more chance (25 times) to a change in placement sequence. With this design, we aim to give chance to the sub-solutions proportional with the number of different combinations obtained in sub-solutions. On the other hand, the worst performing method is method 4 which is making a change in both sub-solutions for determining a neighbor. This result may be expected because a good solution may turn into a very bad solution with a change in both sub-solutions.

ABC meta-heuristic has three parameters each having three values. This makes a total of 27 different combinations. Additionally, *nct* has five values and thus the number of combinations increases to 135. The performance analysis results for the parameters are given in Table 5 where the percentage deviations from the overall best solution found by ABC or SA are displayed.

We firstly observe that increasing the value of *noc* parameter, the assembly time decreases. This is because of the fact that a high value for the *noc* parameter gives the opportunity to look on broader solutions. *limit* parameter determines the time to abandon a solution and replace it with a new one through scout bees. As we pointed out in previous section, if the *limit* value is small, the improvement possibilities of a solution can be left unfinished and it is replaced with new a solution. On the other hand, an excessively high *limit* value causes the algorithm to wander about the same solutions' neighbors. This phenomenon is exactly observed with the results and we decide to use 1000 for the value of the *limit* parameter. The results on *cs* parameter points out that there is no effect of different values of *cs* on the assembly time and we decided to use 20 as the *cs* value. Lastly, for the *nct* parameter, except method 4, there is little difference among the other values. Thus, as SA uses method 2 for *nct*, we prefer to use the same technique for ABC.

5.3. Results on real PCBs

The results of iterative methods on real PCBs together with the meta-heuristic approach for the master problem are given in Table 6.

Results given in Table 6 indicate that iterative methods outperform the meta-heuristic approaches with a cost of high run time. However, in PCB production environments because the production volume is large, a small percentage of improvement in PCB assembly increases productivity and brings huge amounts of profit to the company. Therefore, better optimized

Table 7
Random PCB results.

Method	Homogeneous		Structured	
	Assembly time	Run time	Assembly time	Run time
Iterative 1	**27.25**	751.81	**26.12**	739.32
Iterative 2	28.79	779.13	27.55	759.25
Iterative 3	29.19	796.89	28.42	782.05
Iterative 4	29.26	777.39	28.44	764.67
SA	30.86 ± 1.35	4.44	30.96 ± 1.46	4.59
ABC	31.99 ± 0.86	8.06	31.93 ± 0.92	8.02

assembly plans are preferred even though they take longer time to build. Between the meta-heuristic approaches, ABC shows inferior performance but its results are comparable with SA. When we analyze the standard deviation values of the algorithms, we observe that ABC is more robust than SA. Due to the high run time complexity of the heuristics exploited to solve the SDTSP and multiple iteration steps embedded in the iterative approach, we run the iterative algorithms only once. Therefore, the iterative methods do not have the standard deviation values.

The superior performance of the iterative approach to the meta-heuristics that solve the combined problem is appreciable. Obviously one of the reasons of this is the quite larger run time. However, when we analyze the results obtained after the first iteration of the iterative method (which needs only one twentieth run time) we see that iterative approach still contains considerably better solutions. In other words even if we equate the run times of both approaches we expect that iterative approach will be more successful. This phenomenon can be explained as follows.

SDTSP is a very complicated problem in that it involves components of different weight categories. This may result in a failure of improvement in most pair-wise exchange trials. Thus, a meta-heuristic approach may fail to improve a given solution even if it is allowed to run for a longer amount of time. On the other hand, the constructive heuristic ATMA and the others are developed considering the special structure of the SDTSP and thus they perform better. This situation may lead us to arrive at two main conclusions. First, even the meta-heuristic approaches are applied to many different problems successfully they fail to perform well for our problem. Secondly we can say that, the heuristics developed for the SDTSP are really successful.

Among the iterative methods, we observe that iterative methods 3 and 4 show better performance than others where Iterative 3 is the best on the average.

5.4. Results on random PCBs

In the previous subsections, the best performing parameters of the SA and ABC meta-heuristics are determined. In this subsection their performance on synthetically generated PCBs are analyzed. Table 7 presents the results. The iterative methods run 10 iterations since in some preliminary experiments it was observed that the best solution could be found within this time.

The results indicate that iterative methods again give better results than meta-heuristic approaches. Thus, we can conclude that developing special heuristics for the problems helps us obtain better solutions. The worse behavior of the meta-heuristic approaches can be explained by their deficiency in grouping heavier components in the placement sequence. The standard deviation values of the algorithms show that ABC is more robust than SA.

On the other hand, the high run time of the iterative method can be decreased by using lower number of iterations or redesigning it in terms of improvement heuristics applied. For example the most important factor effecting the run time is the RRTLEM heuristic. Taking out the RRTLEM heuristic or decreasing the iteration number will definitely reduce run time but still may produce favorable results.

Among the iterative approaches, in contrast to the real PCB results, we observe that iterative method 1 shows the best performance followed by method 2. When results on real PCBs are taken into account where method 3 was the winner, we can conclude that considering the PCB at hand each strategy must be tried in order to obtain best assembly time.

5.5. Exploitation of metaheuristics in the iterative method

Since the iterative method outperforms the metaheuristics which tackle on the master problem, we designed some other iterative algorithms that exploit the metaheuristics in the iterative steps and we performed some further computational experiments. In our original iterative method we used heuristics for the SDTSP and SA for the QAP. As a different solution method, we used SA and ABC to solve the subproblems iteratively. Actually, this approach creates four different methods: solving SDTSP with SA and solving QAP with ABC, solving SDTSP with ABC and solving QAP with SA, solving both subproblems with SA, solving both subproblems with ABC. As explained in Section 5.1, in the iterative process there are four methods that can be applied: after SDTSP is solved, QAP can be solved either based on QAP cost or total assembly cost and QAP can start either from a new solution or from the best solution found in the previous iteration. Considering these four iterative methods and four solution methods, it makes 16 combinations to try out. Of those 16 cases, because ABC is a swarm optimization technique, if ABC starts with a solution from previous iteration (i.e. for iterative methods 1 and 2), only one of the initial

Table 8
Assembly time results on real PCBs when metaheuristics are exploited in the iterative method.

Iter. No	Heur. + SA	SA + SA	ABC + SA	SA + ABC	ABC + ABC
Iter.1	51.01	54.18 ± 1.05	60.82 ± 1.37	53.62 ± 0.84	59.12 ± 1.57
Iter.2	51.90	53.99 ± 1.06	59.81 ± 1.72	**53.34** ± 0.99	59.67 ± 1.54
Iter.3	**49.58**	57.47 ± 0.76	62.14 ± 0.99	57.27 ± 0.88	62.24 ± 0.81
Iter.4	49.65	57.24 ± 0.74	62.04 ± 0.84	57.21 ± 0.84	61.88 ± 1.04

Table 9
Assembly time results on random PCBs when metaheuristics are exploited in the iterative method.

Board type	Iter. no	Heur. + SA	SA + SA	ABC + SA	SA + ABC	ABC + ABC
Hom. Boards	Iter.1	**27.25**	31.84 ± 0.80	32.93 ± 0.66	31.48 ± 0.79	31.64 ± 0.89
	Iter.2	28.79	31.50 ± 0.82	32.11 ± 0.91	**31.16** ± 0.88	31.89 ± 0.82
	Iter.3	29.19	32.55 ± 0.70	33.43 ± 0.56	32.38 ± 0.71	33.37 ± 0.59
	Iter.4	29.26	32.64 ± 0.76	33.45 ± 0.60	32.46 ± 0.71	33.34 ± 0.60
Str. Boards	Iter.1	**26.12**	32.09 ± 0.86	33.17 ± 0.70	31.62 ± 0.82	31.69 ± 0.94
	Iter.2	27.55	31.70 ± 0.89	32.23 ± 1.02	**31.18** ± 0.97	31.93 ± 0.92
	Iter.3	28.42	32.85 ± 0.77	33.71 ± 0.61	32.73 ± 0.75	33.60 ± 0.60
	Iter.4	28.44	32.91 ± 0.78	33.73 ± 0.61	32.80 ± 0.72	33.59 ± 0.61

particles (bees) in ABC are set as the best solution from the other algorithm. The metaheuristics are run with the parameter values obtained in previous sections.

Table 8 presents the results of the aforementioned iterative combinations of metaheuristics on real PCBs. In the column headers, a combination of metaheuristics denotes that SDTSP is solved by the first metaheuristic and QAP is solved by the second. That is, ABC + SA means ABC is used for solving SDTSP and SA is used for solving QAP. The results of the metaheuristic combinations are compared with the Heur. + SA combination whose results are provided in Table 6 in detail. Run time values of the computational tests are not provided in detail for brevity, but on the average it takes 67, 97, 82 and 133 s for the SA + SA, ABC + SA, SA + ABC and ABC + ABC combinations, respectively. In Table 8, we provide the average values for the real PCBS. The results present that using SA for SDTSP and using ABC for QAP gives the best performance. Even though its performance is inferior when compared with Heur. + SA combination, its run time is much smaller. The worst performance belongs to ABC + ABC combination although it uses the most time. This can be due to its swarm based design where exploration of the search space is more prioritized than exploitation. Among the iterative methods applied on metaheuristic combinations, we observe that iterative method 2 shows better performance than others.

Table 9 presents the results of the aforementioned iterative combinations of metaheuristics on random PCBs. Associated run time values for these computational tests are 33, 52, 45 and 62 s for the SA + SA, ABC + SA, SA + ABC and ABC + ABC combinations, respectively. We observe that SA + ABC gives the best result among all combinations both for homogeneous boards and structured boards. On the other hand, we observe that iterative method 2 shows better performance than others. Since these facts were also observed in Table 8, we can easily conclude that SA is better for SDTSP and ABC is better for QAP and this combination of metaheuristics work better than their own effort.

6. Summary and conclusion

In this study, assembly time minimization problem emerging from the operations of chip shooter placement machines is analyzed. This problem and its subproblems, namely the placement sequencing and the feeder configuration problems, are stated and formulated explicitly and completely (without simplifying assumptions). Former one is an instance of SDTSP whereas the latter one is an instance of QAP. An iterative method is proposed to solve the problem and is compared with two different meta-heuristics, namely simulated annealing and artificial bee colony which are applied to the combined master problem.

Iterative method is composed of solving the sub-problems iteratively and each sub-problem takes the solution of the other as an input. The methods for solving the placement sequencing problem are taken from former studies, while the feeder configuration method is solved using simulated annealing meta-heuristic. Four different variations of the iterative method are studied and their performances are demonstrated on computational experiments.

The solution methods are compared on both real PCB data and synthetically generated PCB data. The results of the iterative method are favorable and promising. Further studies should focus on decreasing its run time and make comparable with the meta-heuristics.

References

[1] R.K. Ahuja, J.B. Orlin, A. Tiwari, A greedy genetic algorithm for the quadratic assignment problem, Comput. Oper. Res. 27 (2000) 917–934.
[2] A.F. Alkaya, Optimizing the operations of electronic component placement machines (Ph.D. thesis), Marmara University, Istanbul, Turkiye, 2009.
[3] A.F. Alkaya, E. Duman, Literature survey of the operation optimization in chip shooter placement machines, in: Proceedings of PICMET'09, August 2–6, Portland, OR, USA, 2009, pp. 3296–3306.

[4] A.F. Alkaya, E. Duman, A new generalization of the traveling salesman problem, Appl. Comput. Math.-Bak 9 (2) (2010) 162–175.
[5] J.C. Ammons, M. Carlyle, G.W. Depuy, K.P. Ellis, L.F. McGinnis, C.A. Tovey, H. Xu, Computer-aided process planning in printed circuit card assembly, IEEE Trans. Compon. Packag. Manuf. Technol. 16 (1993) 370–376.
[6] M. Ayob, G. Kendall, A survey of surface mount device placement machine optimisation: machine classification, European J. Oper. Res. 186 (2008) 893–914.
[7] E.K. Burke, P.I. Cowling, R. Keuthen, New models and heuristics for component placement in printed circuit board assembly, in: Proceedings of the International Conference on Information Intelligence and Systems, 1999.
[8] E.K. Burke, P.I. Cowling, R. Keuthen, Effective heuristic and metaheuristic approaches to optimize component placement in printed circuit board assembly, in: Proceedings of the Congress on Evolutionary Computation, 2000, pp. 301–308.
[9] L. Chwif, M.R.P. Barretto, L.A. Moscato, A solution to the facility layout problem using simulated annealing, Comput. Ind. 36 (1998) 125–132.
[10] C.C. Chyu, W.S. Chang, A genetic-based algorithm for the operational sequence of a high speed chip placement machine, Int. J. Adv. Manuf. Technol. 36 (2008) 918–926.
[11] Y. Crama, O.E. Flippo, J.V.D. Klundert, F.C.R. Spieksma, The assembly of printed circuit boards: a case with multiple machines and multiple board types, European J. Oper. Res. 98 (1997) 457–472.
[12] Y. Crama, J.V.D. Klundert, F.C.R. Spieksma, Production planning problems in printed circuit board assembly, Discrete Appl. Math. 123 (2002) 339–361.
[13] P. Csaszar, T.M. Tirpak, P.C. Nelson, Optimization of a highspeed placement machine using tabu search algorithms, Ann. Oper. Res. 96 (1–4) (2000) 125–147.
[14] Z. Drezner, Extensive experiments with hybrid genetic algorithms for the solution of the quadratic assignment problem, Comput. Oper. Res. 35 (2008) 717–736.
[15] G. Dueck, New optimization heuristics: the great deluge algorithm and the record-to-record travel, J. Comput. Phys. 104 (1993) 86–92.
[16] E. Duman, Optimization issues in automated assembly of printed circuit boards (Ph.D. thesis), Bogazici University, Istanbul, Turkiye, 1998.
[17] E. Duman, Modelling the operations of a component placement machine with rotational turret and stationary component magazine, J. Oper. Res. Soc. 58 (2007) 317–325.
[18] E. Duman, An application of the multiple TSP in printed circuit board assembly, J. Oper. Logist. 2 (3) (2009) III.1–III.9.
[19] E. Duman, I. Or, Precedence constrained TSP arising in printed circuit board assembly, Int. J. Prod. Res. 42 (2004) 67–78.
[20] E. Duman, I. Or, The quadratic assignment problem in the context of the printed circuit board assembly process, Comput. Oper. Res. 34 (2007) 163–179.
[21] E. Duman, M. Uysal, A.F. Alkaya, Migrating birds optimization: a new metaheuristic approach and its performance on quadratic assignment problem, Inform. Sci. (2012) http://dx.doi.org/10.1016/j.ins.2012.06.032.
[22] K.P. Ellis, F.J. Vittes, J.E. Kobza, Optimizing the performance of a surface mount placement machine, IEEE Trans. Electron. Packag. 24 (2001) 160–170.
[23] S. Grotzinger, Feeder assignment models for concurrent placement machines, IIE Trans. 24 (4) (1992) 31–47.
[24] M. Grunow, H.O. Gunther, A. Fohrenbach, Simulation-based performance analysis and optimization of electronics assembly equipment, Int. J. Prod. Res. 38 (2000) 4247–4259.
[25] M. Grunow, H.O. Günther, M. Schleusener, I.O. Yilmaz, Operations planning for collect-and-place machines in PCB assembly, Comput. Ind. Eng. 47 (2004) 409–429.
[26] C.S. Hardas, T.L. Doolen, D.H. Jensen, Development of a genetic algorithm for component placement sequence optimization in printed circuit board assembly, Comput. Ind. Eng. 55 (2008) 165–182.
[27] W. Ho, P. Ji, Component scheduling for chip shooter machines: a hybrid genetic algorithm approach, Comput. Oper. Res. 30 (2003) 2175–2189.
[28] W. Ho, P. Ji, Optimal Production Planning for PCB Assembly, Springer Series in Advanced Manufacturing, Springer-Verlag London Limited, 2007.
[29] W. Ho, P. Ji, An integrated scheduling problem of PCB components on sequential pick-and-place machines: mathematical models and heuristic solutions, Expert Syst. Appl. 36 (2009) 7002–7010.
[30] W. Ho, P. Ji, Y. Wu, A heuristic approach for component scheduling on a high-speed PCB assembly machine, Prod. Plan. Control 18 (8) (2007) 655–665.
[31] P. Ji, Y.F. Wan, Planning for printed circuit board assembly: the state-of-the-art review, Int. J. Comput. Appl. Technol. 14 (4–6) (2001) 136–144.
[32] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, J. Global Optim. 39 (3) (2007) 459–471.
[33] L.P. Khoo, T.K. Ng, A genetic algorithm-based planning system for PCB component placement, Int. J. Prod. Econ. 54 (1998) 321–332.
[34] B.M. Kiyicigi, E. Duman, A.F. Alkaya, Finding best performing solution algorithm for the QAP, in: Proceedings of IMS2010, Sarajevo, Bosnia Herzegovina, September 15–17, 2010, pp. 510–522.
[35] T. Leipala, O. Nevalainen, Optimization of the movement of a component placement machine, European J. Oper. Res. 38 (1989) 167–177.
[36] E.M. Loiola, N.M. Abreu, P.O.B. Netto, P. Hahn, T. Querido, A survey of the quadratic assignment problem, European J. Oper. Res. 176 (2) (2007) 657–690.
[37] L.F. McGinnis, J.C. Ammons, M. Carlyle, L. Cranmer, G.W. Depuy, K.P. Ellis, C.A. Tovey, H. Xu, Automated process planning for printed circuit card assembly, IIE Trans. 24 (1992) 18–29.
[38] K. Ohno, Z. Jin, S.E. Elmaghraby, An optimal assembly mode of multi-type printed circuit boards, Comput. Ind. Eng. 36 (1999) 451–471.
[39] N.S. Ong, W.C. Tan, Sequence placement planning for high-speed PCB assembly machine, Integr. Manuf. Syst. 13 (2002) 35–46.
[40] R. Sherman, Strong Markets for ICs and their Packaging. New Venture Research. http://www.newventureresearch.com/a-new-advanced-ic-packaging-report/strong-markets-for-ics-and-their-packaging/ (accessed 15.01.13).
[41] J. Smed, M. Johnsson, O. Nevalainen, A hierarchical classification scheme for electronics assembly problems, in: Proceedings of TOOLMET Symposium—Tool Environments and Developments Methods for Intelligent Systems, Oulu, Finland, 2000, pp. 116–119.
[42] T.M. Tirpak, Design to manufacturing information management for electronics assembly, Int. J. Flexible Manuf. Syst. 12 (2–3) (2000) 189–205.
[43] W. Wang, P.C. Nelson, T.M. Tirpak, Optimization of high-speed multistation SMT placement machines using evolutionary algorithms, IEEE Trans. Electron. Packag. 22 (1999) 137–146.
[44] Y. Wu, P. Ji, A scheduling problem for PCB assembly: a case with multiple lines, Int. J. Adv. Manuf. 43 (2009) 1189–1201.
[45] S.H. Yeo, C.W. Low, K.H. Yong, A rule-based frame system for concurrent assembly machines, Int. J. Adv. Manuf. Technol. 12 (1996) 370–376.