Computers & Industrial Engineering 133 (2019) 186–194

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

Printed circuit board assembly time minimisation using a novel Bees Algorithm

Marco Castellani[a,*], Sameh Otri[b], Duc Truong Pham[a]

[a] Department of Mechanical Engineering, University of Birmingham, Birmingham B15 2TT, UK
[b] School of Engineering, Cardiff University, Cardiff CF24 3AA, UK

ARTICLE INFO

Keywords:
Combinatorial optimisation
Bees Algorithm
Printed circuit board assembly
Manufacturing

ABSTRACT

This paper presents a novel version of the Bees Algorithm customised to solve combinatorial optimisation problems. This version was created to minimise assembly time in the manufacturing of printed circuit boards using a machine of the moving-board-with-time-delay type, and optimising the feeder arrangement and machine component placement sequence. The local search procedure of the standard Bees Algorithm was modified to include five new operators for combinatorial optimisation. The customised Bees Algorithm was first tested on the related travelling salesman problem, where it excelled in terms of performance and efficiency compared to three state-of-the-art optimisation methods. It was then applied to a well-known moving-board-with-time-delay benchmark problem, where it performed favourably in comparison to the state-of-the-art in the literature, achieving fast and consistent solutions.

1. Introduction

Printed Circuit Boards (PCBs) are widely used in three major industrial sectors: computers, telecommunications and consumer electronics (Crama, van de Klundert, & Spieksma, 2002). The world market for this kind of components was estimated to exceed $63 billion in 2016, and is expected to reach nearly $74 billion in 2022 (BCC Research, 2016).

PCB assembly is the process of placing electronic components (resistors, capacitors, transistors) of different shapes and sizes at specific locations on the bare board. It is performed automatically employing various types of surface mount technology (SMT) placement machines. These machines are capable of fast component placement, and can handle high and rapid production demands. Despite the speed of the SMT placement machines, assembly is to date one of the most time consuming stages in PCB manufacturing.

The main challenge in PCB assembly is the optimisation of the sequence of placement of the components, in order to minimise the total manufacturing time. This is an NP-complete combinatorial problem which was shown to be akin to the quadratic assignment (feeder arrangement) and travelling salesman (TSP) (component placement sequencing) problems (Alkaya & Duman, 2015; Khoo & Ng, 1998; Khoo & Ng, 1998)). In this kind of problems, the computational solution time using any known algorithm grows in non-polynomial fashion with the number of elements (feeders and electronic components in the PCB).

Intelligent optimisation methods (Pham & Karaboga, 2000) are known to provide satisfactory solutions to complex tasks such as NP-complete combinatorial problems. They have been used by several authors to achieve PCB assembly time minimisation, including Genetic Algorithms (GAs) (Fogel, 2000), Simulated Annealing (Kirkpatrick, Gelatt, & Vecchi, 1983), and Evolutionary Programming (EP) (Nelson & Wille, 1995).

The Bees Algorithm (Pham & Castellani, 2009; Pham et al., 2006) is a popular intelligent technique that was proven effective on combinatorial (Pham, Koc, Lee, & Phrueksanant, 2007) as well as continuous optimisation problems (Pham & Castellani, 2013, 2015). This paper focuses on the application of the Bees Algorithm to solve the Moving Board with Time Delay (MBTD) problem, where the goal is to optimise the placement time for a high speed turret-head chipshooter. On this kind of problem an early version of the Bees Algorithm showed great promise (Pham, Otri, & Darwish, 2007). In this paper, a new problem-specific implementation of the Bees Algorithm with five new operators for combinatorial search is employed.

The performance of the new Bees Algorithm implementation is first tested on the TSP. The TSP is a widely used combinatorial optimisation benchmark, and as such provides a familiar test case for the proposed algorithm. Moreover, the component placement sequence optimisation problem turns out to be a TSP variant (Alkaya & Duman, 2015; Khoo &

* Corresponding author.
E-mail address: M.Castellani@bham.ac.uk (M. Castellani).

https://doi.org/10.1016/j.cie.2019.05.015
Received 15 January 2019; Received in revised form 14 April 2019; Accepted 10 May 2019
Available online 11 May 2019

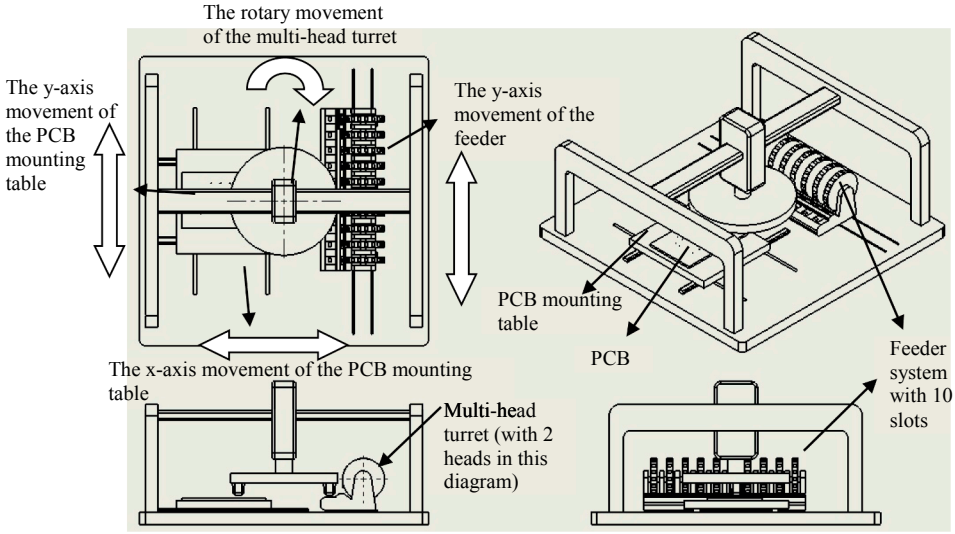


Fig. 1. PCB assembly machine of the MBTD type (with 2 Rotary Turret Heads, 10 Feeder Slots and a Moveable Assembly Table).

Ng, 1998). The new Bees algorithm was compared to the performance of three state-of-the-art heuristic optimisers on the TSP, and then evaluated on the MBTD problem.

Section 2 presents a review of the literature on the domain. Section 3 describes the PCB assembly problem. Section 4 describes the new version of the Bees Algorithm. The proposed algorithm is first tested on the TSP (Section 5), and then on a MBTD benchmark assembly problem (Section 6). Section 7 concludes the paper.

2. Literature review

A variety of intelligent techniques were investigated to minimise PCB assembly time. These methods include Evolutionary Algorithms (Wong & Leu, 1993; EAs) (Nelson & Wille, 1995; Maimon and Brha, 1998; Ong and Khoo, 1999; Ho & Ji, 2007), Particle Swarm Optimisation (PSO) (Hsu, 2017), minimal spanning tree optimisation (Leipälä and Nevalainen, 1989), integer programming (Li & Yoon, 2017; Seth, Klabjan, & Ferreira, 2016), Tabu Search (Luo, Wang, & Hu, 2016), and rule-based expert systems (Yeo, Low, & Yong, 1996), as well as combinations of different optimisation algorithms (Alkaya & Duman, 2015; Han & Seo, 2017; Luo, Liu, & Hu, 2017; Zhu, Ju, & Zhang, 2018). For an overview of intelligent techniques for the general problem of assembly line planning the reader is referred to Rashid, Hutabarat, and Tiwari's (2012) review.

EAs are popular problem solvers due to their simple implementation and robustness against local fitness optima. Amongst the various kinds of EAs, Genetic Algorithms (GAs) were successfully employed by Wong and Leu (1993) and Ong and Khoo (1999) to solve the Moving Board with Time Delay (MBTD) problem. To optimise component placement and feeder assignment, Wong and Leu employed four genetic operators, whilst Ong and Khoo used crossover and a combination of mutation operators. It was shown that Ong's and Khoo's method is easily adaptable to many other types of assembly machine planning problems. Ong and Tan (2002) used a GA to solve the MBTD problem for a high-speed PCB assembly machine, employing eight genetic operators (four crossover operators and four mutation operators).

Swarm Intelligence (Bonabeau, Dorigo, & Theraulaz, 1999) was successfully applied to several combinatorial optimisation problems. Perhaps the first use of this kind of algorithms on combinatorial problems was Dorigo's and Gambardella's (1997) solution of the TSP using Ant Colony Optimisation (ACO). The first application of Swarm Intelligence to the PCB assembly planning problem was Pham, Otri, et al.'s (2007) implementation of the Bees Algorithm. Tested on a benchmark MBTD task, the Bees Algorithm obtained a significant reduction in assembly time compared to GA and EP optimisers (Otri, 2011; Pham, Otri, et al., 2007). Ang, Pham, and Ng (2009) reported good results on the MBTD task from hybridisation of the Bees Algorithm with various operators created from Theory of Inventive Problem Solving (TRIZ) principles (Sheng & Kok-Soo, 2010). However, the TRIZ-enhanced Bees Algorithm used some form of population seeding, which brought the initial population already close to the found optimum.

Hsu (2017) applied a customised version of PSO, and found it outperformed a GA optimiser. Alkaya and Duman (2015) tested different combinations of optimisers, and obtained the best results iterating cycles of SA for component placement search and ABC for feeder layout optimisation.

In addition, intelligent techniques like EAs and Artificial Bee Colony optimisation were applied to time minimisation problems involving different PCB assembly devices like multi-head gantry machines (Guo, Geng, Takahashi, Wang, & Jin, 2018; Lin & Huang, 2017) or a Kuka robot (Andrzejewski, Cooper, Griffiths, & Giannetti, 2018).

In the following sections of the paper, the performance of the new Bees Algorithm implementation will be compared with the performance of three different kinds of EAs (Leu, Wong, & Ji, 1993; Nelson & Wille, 1995; Ong and Tan, 2002), one hybrid GA (Ho & Ji, 2007), two implementations of the standard Bees Algorithm (Pham, Otri, et al., 2007), and one Bees Algorithm implementation enhanced with TRIZ operators (Ang, Ng, Pham, & Soroka, 2013).

3. The printed circuit board assembly optimisation problem

This section describes the MBTD machine, and the assembly sequence optimisation problem.

3.1. The PCB moving board with the time delay (MBTD) assembly machine

There are three types of machines for placing through-hole and surface-mount components in sequence onto a PCB (Pham, Otri, et al., 2007). The most complex type is considered in this study: the MBTD assembly machine (Fig. 1). It comprises three moving parts:

- An array of feeders that moves along a single axis (y). It brings the feeder with the required component to the fixed pick-up location for assembly. The pickup location is the centre of the array of feeders.
- An assembly fixed-axis multi-head rotating turret. It picks up components from the feeder array with one head, whilst one of the other head(s) simultaneously places another component onto the PCB board. After that, the turret rotates to pick up and deploy new

components. The axis of rotation of the turret is fixed to the z axis.
- A moving $x-y$ table which carries the PCB. The table moves to bring the PCB to the fixed component placement location, in accordance with the assembly sequence and location of the components.

Component pick-up and placement occur simultaneously after the correct feeder and the table have reached their designated positions, and the turret has completed indexing the appropriate pick-up and placement heads (Ayob, Cowling, & Kendall, 2002). The MBTD machine can also place components of different types. In order to perform the task, the multiple feeders, the multi-head turret pick-and-place system, and the assembly table need to be synchronised.

Since there are three moving parts in this type of machine (the board, feeder, and turret), each of these three parts has to wait that the other two complete their movement, before the next component can be picked up or placed. Hence, the time T_i needed for the placement of component i is the maximum between the times needed for the board movement, feeder movement, and indexing. The main difficulty in minimising the total assembly sequence time is that two optimisation problems need to be addressed concurrently (Ho & Ji, 2005):

- The placement sequence of the board components needs to be determined
- The various types of components need to be assigned to the feeders

3.2. Printed circuit board assembly planning

PCB assembly planning involves two tasks: set-up management and process optimisation. The ultimate goal is to reduce assembly cycle times.

In the context of planning for a single assembly machine with an array of feeders, set-up management may include the solution of the component allocation problem: that is, the arrangement of the allocation of components among the different feeders. Alternatively, the relative position of the feeders in the array needs to be optimised (feeder arrangement problem). In this study, feeder arrangement will be included in the set-up management task. Henceforth, a given arrangement of R feeders will be denoted as a sequence $F=\{f_1, f_2, \cdots, f_R\}$, where f_j is the j^{th} feeder in the sequence. Each feeder f_j is instantiated to a different integer number (label) $r \in [1, R]$, representing the r^{th} type of component. The goal of the arrangement task is to minimise feeder movements, and hence component pick-up time.

Process optimisation consists of finding the optimal sequence of placement for the components on the board. It is essentially a component sequencing problem. Henceforth, the sequence of placement of components will be denoted as $C=\{c_1, c_2, \cdots, c_N\}$, where c_i is the i^{th} component that is placed. Each c_i is instantiated to a different integer number (label) $p \in [1, N]$, representing the p^{th} location on the board where the component has to be placed. The aim of process optimisation is to minimise the movements of the table, and hence placement time.

3.3. Formalisation of the assembly problem

As discussed above, in an MBTD machine the assembly time is affected by three factors. These are the movement of the PCB, the shifting time of the turret head, and the travelling time of the feeder carrier. The total assembly time needed for a PCB is the summation of the dominating times associated with these three factors for all board components. The formula for the total assembly time T_{Total} is defined as follows (Ho & Ji, 2007).

$$T_{Total} = \sum_{i=1}^{N+\frac{h}{2}} T_i \tag{1}$$

where N is the total number of components to be placed onto the PCB, h is the total number of assembly heads on the turret, and T_i is the time required to place component c_i. T_i depends on three terms, each one related to a different operation. The largest of the three terms (longest time) determines T_i.

$$T_i = \max\left[t_1(c_{i-1}, c_i),\ t_2\left(f_{i+\frac{h}{2}-1}, f_{i+\frac{h}{2}}\right),\ t_3\right] \tag{2}$$

The first term $t_1(c_{i-1}, c_i)$ is the time required to move the table from the location of component c_{i-1} to the location (x_i, y_i) of componentc_i. It is given by the Chebyshev metric (Ho & Ji, 2007):

$$t_1(c_{i-1}, c_i) = \max\left(\frac{x_{i-1}-x_i}{v_x}, \frac{y_{i-1}-y_i}{v_y}\right) \tag{3}$$

where $v_f=(v_x, v_y)$ is the velocity of the table ($x-y$ plane) which brings the PCB to the fixed component placement location. The Chebyshev metric essentially takes into consideration the x and y movements of the board as independent motions. This is usually the case when the table is controlled by two motors, one for each axis. In this study, the starting position c_0 of the table is set at the origin(0, 0) of the Cartesian frame of reference, and the maximum v_x and v_y velocities along the two axes are fixed to $60\cdot10^{-3}$ ms^{-1}.

The second term $t_2\left(f_{i+\frac{h}{2}-1}, f_{i+\frac{h}{2}}\right)$ is the time needed to move the feeder array from the slot where component $c_{i+\frac{h}{2}-1}$ is supplied, to the slot where component $c_{i+\frac{h}{2}}$ is supplied. Component $c_{i+\frac{h}{2}}$ is picked up when component c_i is placed onto the PCB. As c_N is the last component to be assembled onto a given PCB, c_{N+h} is the h^{th} component to be placed onto the next PCB. The component pickups are considered to be at the centre of the feeders. The distance between feeders is measured using the Euclidean metric (Ho & Ji, 2007).

$$t_2(f_i, f_j) = \frac{\sqrt{(x_i^f - x_j^f)^2 + (y_i^f - y_j^f)^2}}{v_f} \tag{4}$$

where (x_i^f, y_i^f) are the Cartesian coordinates of feeder f_i and $|v_f| = \sqrt{v_x^2 + v_y^2}$ is the norm of the table velocity vector. For the case studied in this paper, the feeders are arranged in a straight line along the y axis (Fig. 1) and separated by a distance of 15 mm. Hence, there will be no movement along the x axis, and Eq. (4) becomes:

$$t_2(f_i, f_j) = \frac{|y_i^f - y_j^f|}{v_f} \tag{5}$$

The third term t_3 is the time taken by the turret to index the assembly heads by one position, and is set to 0.25 s per step. Generally, indexing takes place one position at a time and always in the same direction.

In this study, the two problems of set-up management and process optimisation will be tackled simultaneously, with the goal of minimising T_{Total}. Formally, the task entails the solution of a combinatorial optimisation problem, where the sequence $\{C, F\} = \{c_1, c_2, \cdots, c_N, f_1, f_2, \cdots, f_R\}$ (Fig. 2) needs to be optimised, in order to minimise $T_{Total} = g(C, F)$. The function $g(C, F)$ depends on the combinatorial arrangement of the values of C and F.

The assembly problem can be visualised through a graph consisting of a finite number of nodes and links between nodes. Each node

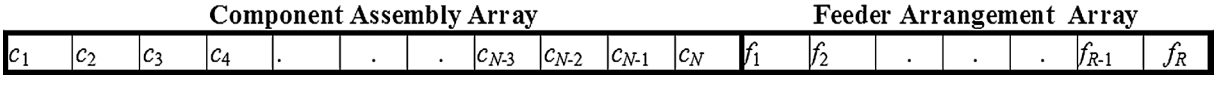


Fig. 2. Representation of a PCB assembly sequence.

represents the position of one of the components on the board. A link between two nodes indicates a transition between the two corresponding sites on the board. It is associated to a cost, representing the travelling time between the two locations. A solution to the PCB assembly problem can be visualised as a complete path (assembly sequence) touching all nodes once and only once, and is associated to the cost T_{Total} (Eq. (1)). The objective of the Bees Algorithm is to find the graph of minimum total cost.

4. The Bees Algorithm for PCB optimisation

Each solution to the PCB assembly problem comprises two concatenated arrays (Fig. 2). The first array represents the sequence of component placements and the second represents the feeder arrangement. A valid feeder arrangement is any of the possible permutations of feeder labels, each label corresponding to a particular feeder and component type. For example, label $r \in [1, R]$ might indicate feeder r which supplies 100 KΩ resistors. The number of labels R in a valid feeder arrangement must be equal to the number of feeders in the assembly machine. A valid placement sequence is also a permutation of labels, each label $p \in [1, N]$ representing a placement position on the PCB. The total number of labels N must be equal to the number of placement positions. The objective of the Bees Algorithm is to optimise the sequence of component placement steps and feeders in order to minimise assembly time.

The Bees Algorithm (Pham & Castellani, 2009) models a colony of honeybees searching for and exploiting multiple food sources (solutions). Initially, a number of scout bees search randomly the space of feasible solutions for good candidates. That is, a number of random solutions are generated and evaluated. A scout is associated to the solution it visited.

Each scout recruits a number of foragers according to the quality of the found solution (fitness = T_{Total}), through a selection procedure inspired by the waggle dance of biological bees (Seeley, 1996). The role of the foragers is to perform local exploitative search: that is, to search in the neighbourhood of the solution visited by the scout. Each neighbourhood in the search space can be visualised as a flower patch in nature, where forager bees are directed by the scouts.

Foragers generate and evaluate solutions similar to those located by the scout. The Bees Algorithm keeps on searching the solution space in the neighbourhood of the best locations, until better solutions are found. The procedure can be summarised as a parallel local search on multiple local patches. When the search stops yielding improvements in a local neighbourhood (the flower patch is exhausted of nectar), the neighbourhood is abandoned and the scout is sent to search a new location (is randomly re-initialised). This procedure prevents the local search from being trapped into sub-optimal peaks of performance. If the abandoned site contained the best-so-far found solution, this solution is saved, and if not bettered by any other local search site result, it will become the final solution of the Bees Algorithm procedure.

Fig. 3 shows the flowchart of Bees Algorithm. Each step is detailed below. The proposed procedure adapts the standard Bees Algorithm described by Pham and Castellani (2009) to the MBTD assembly problem.

4.1. The algorithm

At start, n scout bees are placed at random with uniform probability in the solution space. That is, n valid solutions (Fig. 2) are randomly generated and evaluated (Eqs. (1)–(4)). The algorithm then enters its main cycle, which is repeated for a given number *itr* of times. The $ne \leq n$ scouts that found the solutions of highest fitness are called 'elite bees'. They recruit the largest number of foragers (*nre*) for neighbourhood search. The remaining scouts recruit $nrb \leq nre$ foragers for local search.

Each of the *nre* (*nrb*) recruited foragers, visits (generates) a solution similar to the one marked by the scout. Five problem-specific operators were designed for neighbourhood search. They all generate valid solutions from permutations of existing feeder arrangement and component placement sequences. For each neighbourhood, the forager that found the best solution is selected as the new scout. If no forager landed on a better solution, the old scout is retained. The five operators are described below for the component placement sequence problem. The same procedures apply for the feeder arrangement problem. That is, the five operators are applied separately to the two arrays representing respectively the component placement sequence and feeder arrangement.

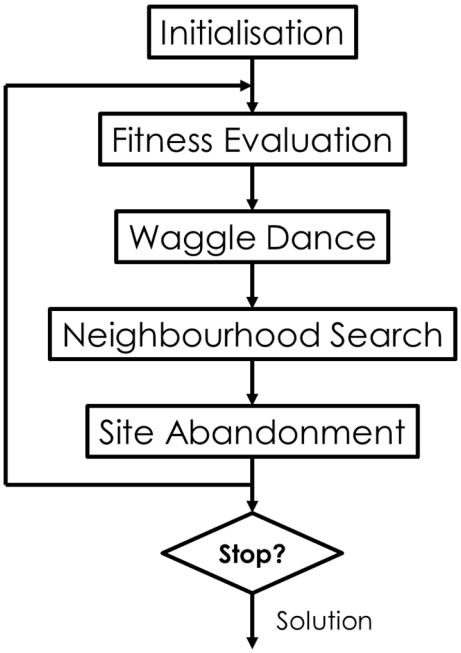


Fig. 3. Flowchart of Bees Algorithm.

During one optimisation cycle, neighbourhood search is performed either on the feeder arrangement or the component placement sequence. That is, feeder arrangement and components placement are optimised in alternate steps. Experimentally, the best results were obtained alternating one cycle of feeder arrangement to five cycles of component placement optimisation.

The progress of the local optimisation procedure is checked in each neighbourhood. If no recruited forager landed on a solution that improved the local best marked by the scout, the search is said to stagnate. After a given number *stlim* of iterations where no local improvement is recorded, the local food source (neighbourhood) is considered exhausted (i.e. the optimum was found) and is abandoned. In this case, the scout is randomly re-initialised.

The five problem-specific operators are described below.

4.1.1. Block insertion operator

The action of this operator is illustrated in the example of Fig. 4, where the placement sequence $\{c_1, c_2, c_3, c_4, c_5, c_6\} = \{3, 2, 5, 1, 4, 6\}$ is modified as follows. The insertion operator picks two randomly chosen positions (c_1 and c_4), and removes the sequence ($\{2, 5\}$) in between. The removed section is then inserted at a randomly selected point within the sequence, in this example after c_5. The re-arranged sequence becomes: $\{c_1, c_2, c_3, c_4, c_5, c_6\} = \{3, 1, 4, 2, 5, 6\}$. The net effect is that a section of the sequence of placement operations is shifted forward or backward in the order (Pham, Otri, et al., 2007).

4.1.2. Single-point insertion operator

This operator works like the block insertion operator, but is limited to one single instead of a sequence of placement operations. Given the

Original sequence :	{3, 2, 5, 1, 4, 6}
Sequence after removal of section {2, 5}:	{3, 1, 4, 6}
Sequence following shift of {2, 5}:	{3, 1, 4, 2, 5, 6}

a) procedure

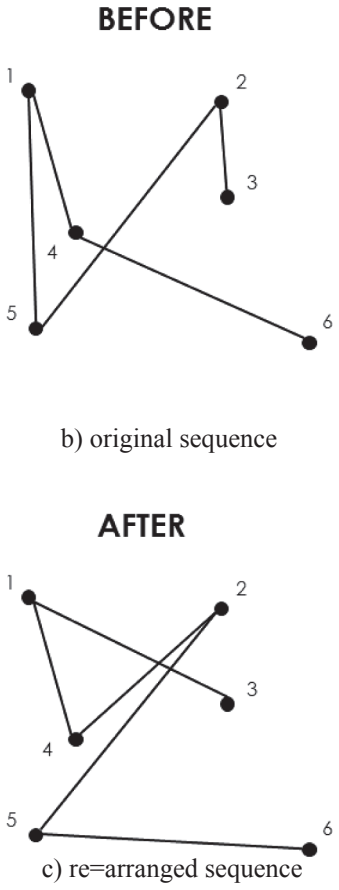


Fig. 4. Block insertion.

sequence $\{c_1, c_2, c_3, c_4, c_5, c_6\} = \{3, 2, 5, 1, 4, 6\}$, the Single-point insertion operator takes one random element (e.g. $c_3 = 5$) and moves it at a randomly selected new position (e.g. after $c_5 = 4$). The new sequence becomes {3, 2, 1, 4, 5, 6}.

4.1.3. The 2-Opt operator

In combinatorial optimisation, edge exchange procedures (Lin & Kernighan, 1973) are amongst the best known and most effective tour improvement methods (Okano, Misono, & Iwano, 1999). The k-paths optimal (k-Opt) algorithm tests exhaustively all feasible exchanges of k edges to improve the current solution; this solution is said to be k-optimal. Since the number of feasible exchanges increases rapidly with the number of edges, k is usually set to 2 or 3.

The 2-Opt operator randomly breaks a solution into two sections. The component placement sequence in one of the sections is reversed, and the two sections are then reconnected. Fig. 5 visualises the procedure for the same sequence {3, 2, 5, 1, 4, 6} of the previous examples. In the figure, the sequence is broken into two sections: {3, 2} and {5, 1, 4, 6}. The second section has its placement sub-sequence inverted ({6, 4, 1, 5}), and is then re-joined to the first to form the new sequence ({3, 2, 6, 1, 4, 5}).

Original sequence :	{3, 2, 5, 1, 4, 6}
Broken-up sequence:	{3, 2} ↔ {5, 1, 4, 6}
Section 1:	{3, 2}
Section 2:	{5, 1, 4, 6}
Section 2 – inverted sub-sequence:	{6, 4, 1, 5}
New sequence (Section 1 + Section 2):	{3, 2, 6, 1, 4, 5}

a) procedure

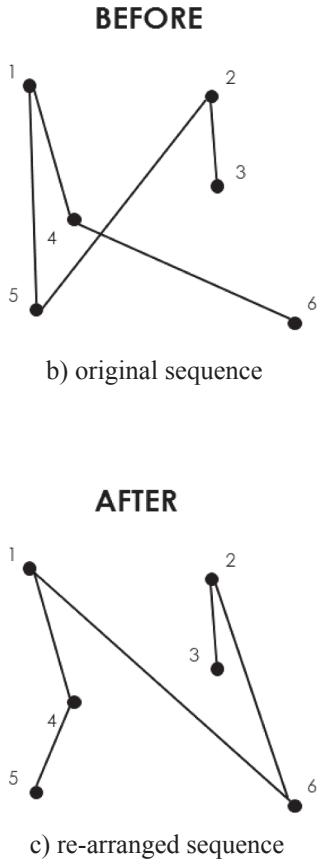


Fig. 5. 2-Opt operator.

4.1.4. Simple Swap operator

This operator takes two random components in the sequence, and swaps their placement order (Lin & Kernighan, 1973). For example, given the sequence $\{c_1, c_2, c_3, c_4, c_5, c_6\} = \{3, 2, 5, 1, 4, 6\}$ and two randomly picked loci $c_3 = 5$ and $c_5 = 4$, the order of placement of components 5 and 4 is swapped. The new sequence becomes {3, 2, 4, 1, 5, 6}.

4.1.5. Neighbour Swap operator

This operator acts like the Simple Swap operator, but in a more localised fashion. It randomly selects two neighbouring elements in the placement sequence, and exchanges their placement order. For example, given the sequence $\{c_1, c_2, c_3, c_4, c_5, c_6\} = \{3, 2, 5, 1, 4, 6\}$ and two randomly picked loci $c_3 = 5$ and $c_4 = 1$, the order of placement of components 5 and 1 is swapped. The new sequence becomes {3, 2, 1, 5, 4, 6}.

4.2. Comparison with other versions of the Bees Algorithm

As outlined above, the proposed algorithm is a customised version of the Bees Algorithm for the MBTD assembly problem. The main adaptation is in the definition of the neighbourhood search operators.

In this case, instead of defining a proximity metric as the neighbourhood size *ngh* (Pham & Castellani, 2009), a number of local operators were defined. These operators change a sequence of components or feeders into a reasonably similar sequence. Some of these operators (2-Opt, Block Insertion) are more disruptive than others (Simple Swap, Neighbours Swap, Single-point Insertion). As a consequence of the lack of a fixed neighbourhood, the neighbourhood shrinking procedure (Pham & Castellani, 2009) has not been implemented.

Another modification of the standard Bees Algorithm is the suppression of the global search procedure. The rationale behind this amendment is the complexity of the solutions, which makes it highly unlikely that randomly generated solutions are able to compete in quality with solutions evolved through local search. Experimental tests confirmed that including additional scouts for global search brings no benefits to the effectiveness of the search.

Pham, Otri, et al. (2007) used a simpler implementation of the standard Bees Algorithm (Pham & Castellani, 2009) which will be henceforth denoted as the Basic Bees Algorithm (BBA). The BBA does not use site abandonment, and does not differentiate between elite and best sites. The BBA used only two search operators: 2-Opt and Single-point Insertion. The two operators were randomly applied either to the feeder arrangement of component placement sequence. Pham, Otri, et al. (2007) tried also to seed each scout of the BBA with a good solution created from one preliminary run of the BBA.

Ang et al. (2009) enhanced the BBA with a number of TRIZ-inspired operators. Henceforth, their algorithm will be called BBA + TRIZ. Also the BBA + TRIZ algorithm was seeded with good initial solutions.

5. Application to the travelling salesman problem (TSP)

The performance of proposed Bees Algorithm was first evaluated on the well-known travelling salesman combinatorial optimisation problem. The TSP was shown by Khoo and Ng (1998) to be akin to the MBTD assembly problem. The performance of the new Bees Algorithm implementation was compared to the performance of three state-of-the-art optimisation methods: Ant Colony Optimisation (ACO), a Genetic Algorithm (GA), and a greedy Nearest Neighbour optimiser (NN).

The population size for ACO, the GA, and the Bees Algorithm was set to 200 individuals, whilst the other parameters were experimentally optimised. To optimise the parameters, each algorithm configuration was tested on a set of 10 randomly generated maps, each comprising 100 cities. On each map, 10 independent optimisation trials were run for each configuration. The configuration that achieved the lowest cumulative distance over the 10 *trials* × 10 *maps* was chosen. Table 1 shows the optimal parameter setting for ACO, the GA, and the Bees Algorithm. Being a greedy deterministic optimiser, the NN algorithm does not need parameter optimisation.

Table 1
Optimal configurations for the TSP benchmark.

Parameter	Symbol	Value
ACO		
Evaporation rate	α	0.2
Distance relative to pheromone weight	β	1
Number of iterations	*itrA*	5000
GA		
Number of iterations	*itrG*	4000
Bees Algorithm		
Elite sites	*ne*	1
Recruited bees for elite e sites	*nre*	100
Best (non elite) sites	*ne*	4
Recruited bees for non elite sites	*nrb*	25
Stagnation limit	*stlim*	50
Number of iterations	*itrB*	3000

5.1. Ant colony optimisation

Dorigo's and Gambardella's (1997) original ACO algorithm was used. The problem is represented as a graph, and artificial ants use a combination of problem-specific (higher probability to take short graph links, viz. move to neighbouring cities) and general purpose (higher probability to follow graph links with high amount of pheromone) heuristics. Two main parameters define the behaviour of ACO: the pheromone evaporation rate $0 < \alpha < 1$ (a high rate favours random exploration of the search space, a low rate favours exploitation of promising areas); and the weight $\beta > 0$ giving the relative importance of the distance versus pheromone heuristics in picking the sequence of cities.

To optimise the algorithm, a full factorial experiment was designed varying the evaporation rate from 0.05 to 0.25 in steps of 0.05 (set to 0.1 in Dorigo & Gambardella, 1997), the weight given to the nearest neighbour heuristics from 1 to 5 in steps of 1 (set to 2 in Dorigo & Gambardella, 1997), and the number of optimisation cycles from 1000 to 5000 in steps of 1000. A total of 125 tests were performed.

5.2. Genetic Algorithm

Kirk's (2014) GA implementation available at MATLAB Central wad used. This GA version is specifically written for TSP optimisation, and uses tournament selection without replacement to select one parent out of four randomly selected individuals. Each parent is duplicated once, and generates other three children out of three problem-specific mutation operators such as genetic flip (the sequence of cities within a randomly selected sub-path is inverted), neighbours swap (see Section 4.1.5), and genetic slide (a sub-path is shifted one position ahead in the sequence). In summary, each evolution step one quarter of the population is selected for reproduction, and each selected individual generates four offspring: one copy of itself, and three mutants each created by a different operator.

For this particular GA implementation, the number of evolution cycles (generations) is the only parameter that needs to be optimised. Five configurations were tested varying the number of generations from 1000 to 5000 in steps of 1000.

5.3. Nearest neighbour

The deterministic NN procedure starts from the two closest cities in the map, and from there moves to the closest city to the first two. The algorithm iteratively moves to the nearest unvisited city to the last added, until all cities have been visited (Johnson & McGeoch, 1997). NN is thus a greedy local optimisation procedure, always choosing the most advantageous move with a local outlook of one step. The strong points of NN are its low computational overheads and ability to find acceptable (albeit rarely optimal) solutions. Like all local optimisation procedures, NN is liable to occasional failures (Gutin, Yeo, & Zverovich, 2002).

5.4. Bees Algorithm

Twenty-five different configurations of the Bees Algorithm implementation described in Section 4 were tested. These configurations comprised different combinations of parameters giving a total colony size of 200 individuals. The number *ne* of elite sites was varied from 1 to 4, the number *nre* of recruited bees for the elite sites was varied from 25 to 100, the number *nb* of best (non elite) sites was varied from 2 to 10, the number *nrb* of recruited bees for the best sites was varied from to 10 to 50, and the stagnation limit *stlim* from 10 to 50 steps. For each of the twenty-five configurations, the number of evolution steps was varied from 1000 to 5000 in steps of 1000. A total of 125 configurations were thus evaluated.

Table 2
TSP optimisation results on 10 randomly generated maps. For each stochastic algorithm, the first (Q1), second (Q2, median value), and third (Q3) quartile of 10 independent optimisation trials are reported. The best (median) result for each map is marked in bold. Results that are not significantly different from the best (Mann-Whitney *U* test at 5% significance level) are shaded in grey.

Map	ACO			GA			Bees Algorithm			NN
	Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q2	Q3	
1	139.7	145.9	147.5	135.0	136.1	136.9	135.4	**135.9**	137.8	164.8
2	134.2	136.6	138.6	128.5	130.5	131.9	127.1	**129.5**	132.2	174.9
3	146.4	147.8	148.9	135.6	136.1	136.6	134.7	**135.4**	136.3	158.2
4	147.8	151.4	152.8	139.6	140.8	141.8	137.8	**139.8**	141.9	170.9
5	140.6	141.5	145.8	132.5	133.5	134.7	132.2	**133.5**	135.0	153.1
6	130.0	131.8	134.1	126.7	127.5	129.2	124.7	**125.7**	126.6	152.1
7	138.6	139.6	143.0	134.1	134.9	140.2	132.8	**133.5**	134.9	148.3
8	146.9	148.0	149.8	133.4	136.5	139.4	131.8	**133.0**	134.7	178.3
9	143.5	145.3	147.9	135.4	136.0	138.7	134.5	**135.6**	136.0	181.1
10	129.4	130.3	132.7	128.8	129.7	130.8	128.4	**129.7**	131.0	153.5
Cumulative	1441.0	1397.1	1417.9	1360.2	1329.5	1341.5	1346.4	1319.3	1331.6	1635.2
Samples	1×10^6			0.8×10^6			0.6×10^6			-

5.5. Experimental results (TSP benchmark)

Using the optimised parameters in Table 1, each algorithm was evaluated on a new set of 10 randomly generated maps of 100 cities. For each map, 10 independent optimisation trials were performed. Table 2 reports for each map the results in terms of first, second (median value), and third quartile of the minimised travelling distances attained in the 10 independent optimisation trials. The best results are highlighted in bold. For each map, the statistical significance of the differences between the results obtained by the three stochastic algorithms was evaluated using Mann-Whitney U-tests at 5% level of significance. In Table 2, results that are not significantly different from the best are shaded in grey. Table 2 reports also the number of fitness evaluations performed by each stochastic algorithm, and the cumulative sum over the 10 maps of the Q1, Q2, and Q3 values. Fig. 6 uses the cumulative Q1, Q2, and Q3 values to visualise the spread and central tendency of the results obtained by the four algorithms. Since NN is deterministic, it is visualised as one unique point.

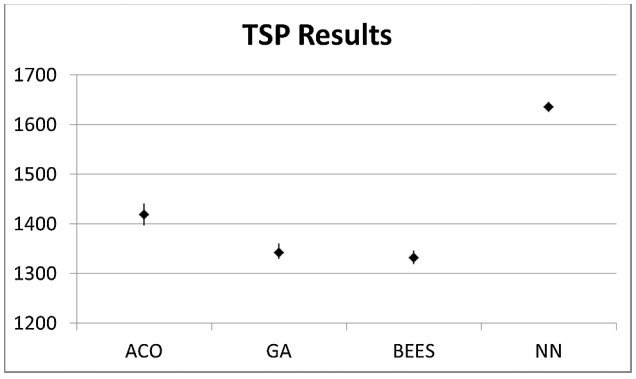


Fig. 6. Cumulative results of TSP optimisation on randomly generated maps.

The results highlight the superiority of the Bees Algorithm and GA over the other two techniques. The Bees Algorithm always obtains the shortest overall path in all 10 maps. The Bees Algorithm is also slightly more consistent than the GA in terms of performance, as indicated by the narrower spread of the results attained (Q3-Q1 difference). However, the current tests show no statistical evidence that the Bees Algorithm is significantly superior to the GA. Increasing the sample size (number of optimisation trials per map) may reveal statistical differences between the two algorithms that are not detectable with the current experimental set up. This is however outside the scope of the paper. The most important result is that the Bees Algorithm needs 25% less function evaluations to attain at least the same results as the GA, which proves the efficiency of the proposed method.

6. Application to the MBTD benchmark problem

The proposed Bees Algorithm is tested on the MBTD benchmark assembly problem described by Leu et al. (1993). The set up consists of 10 feeders, each supplying a different type of component. A total of 50 components need to be placed at fixed positions on the PCB. The coordinates of the placement positions and the parameters of the

Table 3
The parameters of the MBTD assembly machine.

Number of components	50
Number of feeders	10
Number of turret heads	2
Indexing time of turret	0.25 s/index
Average PCB mounting table speed	60 mm/s
Average feeder system speed	60 mm/s
Distance between feeders	15 mm

Table 4
The Parameters of the Bees Algorithm for the MBTD problem

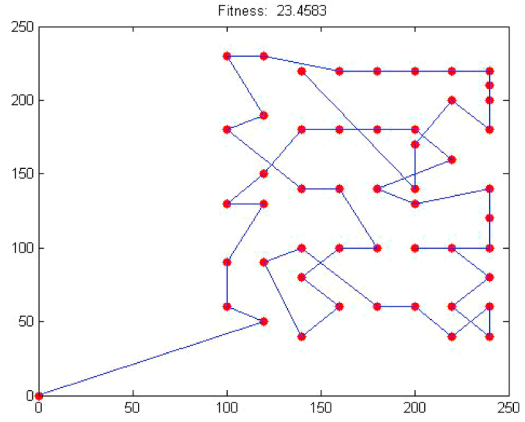
Bees Algorithm parameters	Symbol	Value
Total colony size	*N*	2000
Number of elite sites	*ne*	4
Number of recruited bees for elite e sites	*nre*	300
Number of best (non elite) sites	*ne*	8
Number of recruited bees for non elite sites	*nrb*	100
Stagnation limit	*stlim*	100
Number of iterations	*itr*	3000

Table 5
Optimisation results for the MBTD benchmark: five-number summary of 100 statistically independent trials. The statistics of the assembly times are given in seconds.

Statistic	Value (s)
Minimum	23.46
First quartile	24.71
Median (second quartile)	24.96
Third quartile	25.13
Maximum	25.63

c1→c10	6	1	2	8	3	9	16	21	26	33
c11→c20	38	25	30	46	45	44	29	37	43	36
c21→c30	41	42	35	28	23	14	7	12	18	13
c31→c40	19	24	20	15	4	10	5	11	22	27
c41→c50	34	40	50	49	48	47	39	32	31	17
f1→f10	10	4	1	6	5	2	3	9	8	7

a) solution



b) path on the PCB board

Fig. 7. Optimal solution (23.46 s assembly time).

assembly machine (speed of movement of the table, assembly heads and indexing time of the turret) are fully described in Leu et al. (1993), and summarised in Table 3.

The parameters of the Bees Algorithm were experimentally optimised testing a total of 192 different configurations, performing for each configuration 10 MBTD optimisation trials. The final parameter setting is shown in Table 4. Once the parameters were fixed, 100 independent runs of the algorithm were executed, and the statistical average and spread calculated.

6.1. Experimental results (MBTD benchmark)

The five-number summary of the optimisation results obtained using the customised Bees Algorithm is presented in Table 5. The table proves the consistent performances of the proposed algorithm. The best solution is given in Fig. 7.

A comparison of the obtained results with the state-of-the-art in the literature is difficult, since different authors used different benchmarks (Hsu [20] and Alkaya [14]) and tailored their solutions to different optimisation problems (e.g. Seth et al., 2016 focused on component placement only). Table 6 reports the best solutions obtained in seven published studies which used the same PCB assembly benchmark, and compares them with the best and average results obtained using the customised Bees Algorithm.

Table 6 shows that the proposed method outperforms all algorithms that did not use a seeding procedure. The results are also competitive with those obtained by the algorithms that use a seeding procedure. Indeed, the proposed method found the solution of minimum assembly time (23.46 s).

Unfortunately, the average and spread of the solutions are not available for the other studies. It is therefore impossible to ascertain whether the figures reported in the literature are the expected values of the distribution of results, or the maxima of several attempts (and hence potential outliers to said distribution). Moreover, the methods that used a seeding procedure (Ho & Ji, 2007; Pham, Otri, et al., 2007; Ang et al., 2013) started the search relatively close to the optimum (see third row of Table 6). In this case, it is difficult to distinguish the merits of the optimisation technique from those of the seeding procedure.

It is also difficult to compare the time complexity of the various algorithms. In most cases, the authors reported the number of fitness evaluations performed by the main routine until the optimum was found, and not the whole duration of the optimisation procedure (possibly including the seeding subroutine). Firstly, the result of one run is not representative of the performance of a stochastic optimisation algorithm. Secondly, optimisation algorithms must be run for a duration that guarantees adequate and consistent results, which may be considerably longer than the time the optimum is found in one 'lucky' run. For the above reasons, the number of fitness evaluations was not compared in Table 6. For the customised Bees Algorithm the total number of fitness evaluations executed in one run is equal to the bee colony size (n = 2000, Table 3) times the duration of the algorithm (itr = 3000 optimisation cycles, Table 3), namely $6{\cdot}10^6$ evaluations. As a term of comparison, Ang et al. (2013) obtained a solution of 23.58 s total assembly time, only slightly worse than the solution obtained by the proposed method (23.46 s assembly time). However, their TRIZ-enhanced Bees Algorithm needed nearly $12{\cdot}10^6$ evaluations to find the optimal solution, and the algorithm ran for a total of more than $23{\cdot}10^6$ evaluations. Therefore, it can be concluded that the proposed version of the Bees Algorithm is able to find top quality solutions at a modest computational cost.

Table 6
MBTD benchmark: comparison between the results obtained using the customised Bees Algorithm (cBA) and the state-of-the-art in the literature. Methods 4, 5, and 6 include an initial seeding procedure. The statistics of the assembly times are given in seconds.

References	1	2	3	4	5	5	6	
Optimisation technique	**GA**	**EP**	**GA**	**HGA**	**BBA**	**BBA + seeding**	**BBA + TRIZ**	**cBA**
Average initial solution	70	n/a	60	28.83	54.59	29	35.5	71.08
Best solution	51.5	36	26.9	25.5	25.92	24.08	23.58	23.46
Average solution	n/a	n/a	n/a	n/a	n/a	n/a	n/a	24.96

1: (Leu et al., 1993).
2: (Nelson & Wille, 1995).
3: (Ong and Tan, 2002).
4: (Ho & Ji, 2007).
5: (Pham, Otri, et al., 2007).
6: (Ang et al., 2013).

7. Conclusions

This paper described the application of a customised version of the Bees Algorithm to the optimisation of pick-up and placement sequences for a PCB assembly machine. The proposed method was first tested against 3 state-of-the-art procedures on the TSP, and then applied to the MBTD assembly benchmark problem.

One the TSP, the Bees Algorithm excelled for performance and efficiency. Out of 10 randomly generated maps each containing 100 cities, the proposed algorithm always found the shortest travelling distance, requiring 25% less function evaluations than the GA, and 40% less function evaluations than ACO.

On the benchmark assembly problem, the proposed algorithm was able to obtain a solution requiring an assembly time of 23.46 s, and average solutions requiring 24.96 s. These results compare favourably with the state-of-the-art in the literature.

It is worth noticing that, differently from many recent examples in the literature, the proposed algorithm does not seed the initial population with 'good' (partly optimised) solutions. On the one hand, solution seeding is likely to speed up the search procedure, and possibly help the algorithm to discover better solutions. On the other hand, solution seeding may bias the algorithm towards sub-optimal minima, particularly those characterised by a large basin of attraction. Future work should ascertain the pros and cons of introducing this kind of initialisation procedure.

References

Alkaya, A. F., & Duman, E. (2015). Combining and solving sequence dependent traveling salesman and quadratic assignment problems in PCB assembly. *Discrete Applied Mathematics, 192*, 2–16.

Andrzejewski, K. T., Cooper, M. P., Griffiths, C. A., & Giannetti, C. (2018). Optimisation process for robotic assembly of electronic components. *The International Journal of Advanced Manufacturing Technology, 99*(9–12), 2523–2535.

Ang, M. C., Ng, K. W., Pham, D. T., & Soroka, A. (2013). Simulations of PCB assembly optimisation based on the Bees Algorithm with TRIZ-inspired operators. *International visual informatics conference (IVIC 2013), Selangor, Malaysia* (pp. 335–346). Cham: Springer.

Ang, M. C., Pham, D. T., & Ng, K. W. (2009). Application of the Bees Algorithm with TRIZ-inspired operators for PCB assembly planning. *Proceedings of 5th virtual international conference on intelligent production machines and systems (IPROMS2009), Cardiff, UK* (pp. 454–459). Elsevier.

Ayob, M., Cowling, P., & Kendall, G. (2002). Optimisation for surface mount placement machines. *Proceedings of IEEE international conference on industrial technology (ICIT'02), Bangkok, Thailand* (pp. 498–503). New York, USA: IEEE Press.

BCC Research (2016). Printed circuit boards: Technologies and global markets. Available online: < www.bccresearch.com/market-research/semiconductor-manufacturing/printed-circuit-boards-tech-markets-report-smc103a.html > [last accessed August 2018].

Bonabeau, E., Dorigo, M., & Theraulaz, G. (1999). *Swarm intelligence: From natural to artificial intelligence.* New York, USA: Oxford University Press.

Crama, Y., van de Klundert, J., & Spieksma, F. C. R. (2002). Production planning problems in printed circuit board assembly. *Discrete Applied Mathematics, 123*(1–3), 339–361.

Dorigo, M., & Gambardella, L. M. (1997). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on evolutionary computation, 1*(1), 53–66.

Fogel, D. B. (2000). *Evolutionary computation: Toward a new philosophy of machine intelligence* (2nd edn.). New York, USA: IEEE Press.

Guo, S., Geng, F., Takahashi, K., Wang, X., & Jin, Z. (2018). A MCVRP-based model for PCB assembly optimisation on the beam-type placement machine. *International Journal of Production Research.* https://doi.org/10.1080/00207543.2018.1555380.

Gutin, G., Yeo, A., & Zverovich, A. (2002). Traveling salesman should not be greedy: Domination analysis of greedy-type heuristics for the TSP. *Discrete Applied Mathematics, 117*(1–3), 81–86.

Han, J., & Seo, Y. (2017). Mechanism to minimise the assembly time with feeder assignment for a multi-headed gantry and high-speed SMT machine. *International Journal of Production Research, 55*(10), 2930–2949.

Ho, W., & Ji, P. (2005). A genetic algorithm to optimise the component placement process in PCB assembly. *International Journal of Advanced Manufacturing Technology, 26*(11–12), 1397–1401.

Ho, W., & Ji, P. (2007). *Optimal production planning for PCB assembly.* London, UK: Springer Science & Business Media.

Hsu, H. P. (2017). Solving feeder assignment and component sequencing problems for printed circuit board assembly using particle swarm optimization. *IEEE Transactions on Automation Science and Engineering, 14*(2), 881–893.

Johnson, D. S., & McGeoch, L. A. (1997). The traveling salesman problem: A case study in local optimization. In E. H. L. Aarts, & J. K. Lenstra (Eds.). *Local search in combinatorial optimization* (pp. 215–310). (1st ed.). London, UK: John Wiley and Sons.

Khoo, L. P., & Ng, T. K. (1998). A genetic algorithm-based planning system for PCB component placement. *International Journal of Production Economics, 54*(3), 321–332.

Kirk, J. (2014). Traveling salesman problem – Genetic Algorithm, version 1.3.0.0. Available at: < https://uk.mathworks.com/matlabcentral/fileexchange/13680-traveling-salesman-problem-genetic-algorithm > [last accessed: Sept 2018].

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220*(4598), 671–680.

Leipälä, T., & Nevalainen, O. (1989). Optimization of the movements of a component placement machine. *European Journal of Operational Research, 38*(2), 167–177.

Leu, M. C., Wong, H., & Ji, Z. (1993). Planning of component placement/insertion sequence and feeder setup in PCB assembly using genetic algorithm. *Transactions of ASME Journal of Electronic Packaging, 115*, 424–432.

Li, D., & Yoon, S. W. (2017). PCB assembly optimization in a single gantry high-speed rotary-head collect-and-place machine. *The International Journal of Advanced Manufacturing Technology, 88*(9–12), 2819–2834.

Lin, C. J., & Huang, M. L. (2017). Modified artificial bee colony algorithm for scheduling optimization for printed circuit board production. *Journal of Manufacturing Systems, 44*, 1–11.

Lin, S., & Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research, 21*(2), 498–516.

Luo, J., Liu, J., & Hu, Y. (2017). An MILP model and a hybrid evolutionary algorithm for integrated operation optimisation of multi-head surface mounting machines in PCB assembly. *International Journal of Production Research, 55*(1), 145–160.

Luo, J., Wang, J., & Hu, Y. (2016). A Tabu search with multiple neighborhoods for an integrated operation optimization in electronic industry. *Proceedings 35th Chinese control conference, Chengdu, CN* (pp. 2825–2830). New York, USA: IEEE Press.

Maimon, O. Z., & Brha, D. (1998). A genetic algorithm approach to scheduling PCBs on a single machine. *International Journal of Production Research, 36*(3), 761–784.

Nelson, K. M., & Wille, L. T. (1995). Comparative study of heuristics for optimal printed circuit board assembly. *Proceedings Southcon, Fort Lauderdale, FL* (pp. 322–327). New York, USA: IEEE Press.

Okano, H., Misono, S., & Iwano, K. (1999). TSP construction heuristics and their relationships to the 2-Opt. *Journal of Heuristics, 1*, 71–88.

Ong, N., & Khoo, L. P. (1999). Genetic algorithm approach in PCB assembly. *Integrated Manufacturing Systems, 10*(5), 256–265.

Ong, N. S., & Tan, W. C. (2002). Sequence placement planning for high-speed PCB assembly machine. *Integrated Manufacturing Systems, 13*(1), 35–46.

Otri, S. (2011). *Improving the Bees Algorithm for complex optimisation problems*PhD Thesis. Cardiff University.

Pham, D. T., & Castellani, M. (2009). The Bees Algorithm – Modelling foraging behaviour to solve continuous optimisation problems. *Proceedings of the Institution of Mechanical Engineers, Part C, 223*(12), 2919–2938.

Pham, D. T., & Castellani, M. (2013). Benchmarking and comparison of nature-inspired population-based continuous optimisation algorithms. *Soft Computing, 18*(5), 871–903.

Pham, D. T., & Castellani, M. (2015). A comparative study of the Bees Algorithm as a tool for function optimisation. *Cogent Engineering, 2*(1).

Pham, D. T., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., & Zaidi, M. (2006). The Bees Algorithm - A novel tool for complex optimisation. *Proceedings of 2nd virtual international conference on intelligent production machines and systems (IPROMS 2006)* (pp. 454–459). Elsevier.

Pham, D. T., & Karaboga, D. (2000). *Intelligent optimisation techniques. Genetic algorithms, tabu search, simulated annealing and neural networks.* Springer New York.

Pham, D. T., Koc, E., Lee, J. Y., & Phrueksanant, J. (2007b). Using the Bees Algorithm to schedule jobs for a machine. *Proceedings eighth international conference laser metrology, CMM, and machine tool performance, Cardiff, UK* (pp. 430–439). Bedford, UK: Euspen.

Pham, D. T., Otri, S., & Darwish, A. H. (2007a). Application of the Bees Algorithm to PCB assembly optimisation. *Proceedings of 3rd virtual international conference on intelligent production machines and systems (IPROMS 2006), Dunbeath, Whittles* (pp. 511–516). .

Rashid, M. F. F., Hutabarat, W., & Tiwari, A. (2012). A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches. *The International Journal of Advanced Manufacturing Technology, 59*(1–4), 335–349.

Seeley, T. D. (1996). *The wisdom of the hive: The social physiology of honey bee colonies.* Cambridge, USA: Harvard University Press.

Seth, A., Klabjan, D., & Ferreira, P. M. (2016). A new novel local search integer-programming-based heuristic for PCB assembly on collect-and-place machines. *Mathematical Programming Computation, 8*(1), 1–45.

Sheng, I. L. S., & Kok-Soo, T. (2010). Eco-efficient product design using theory of inventive problem solving (TRIZ) principles. *American Journal of Applied Sciences, 7*(6), 852–858.

Wong, H., & Leu, M. C. (1993). Adaptive genetic algorithm for optimal printed circuit board assembly planning. *Annals CIRP, 42*(1), 17–20.

Yeo, S. H., Low, C. W., & Yong, K. H. (1996). A rule-based frame system for concurrent assembly machines. *The International Journal of Advanced Manufacturing Technology, 12*(5), 370–376.

Zhu, G. Y., Ju, X. W., & Zhang, W. B. (2018). Multi-objective sequence optimization of PCB component assembly with GA based on the discrete Fréchet distance. *International Journal of Production Research, 56*(11), 1–18.