This paper was accepted for publication in *International Journal of Production Research*
http://dx.doi.org/10.1080/00207543.2016.1200154

An MILP model and a hybrid evolutionary algorithm for integrated operation optimization of multi-head surface mounting machines in PCB assembly

Jiaxiang Luo[a]*, Jiyin Liu[b] and Yueming Hu[a]

[a] School of Automation Science and Engineering,
South China University of Technology, Guangzhou, China;

[b] School of Business and Economics,
Loughborough University, Leicestershire LE11 2UA, UK

Abstract:

This paper focuses on an operation optimization problem for a class of multi-head surface mounting machines in printed circuit board assembly lines. The problem involves five interrelated sub-problems: assigning nozzle types as well as components to heads, assigning feeders to slots and determining component pick-up and placement sequences. According to the depth of making decisions, the sub-problems are first classified into two layers. Based on the classification, a two-stage mixed integer linear programming (MILP) is developed to describe it and a two-stage problem-solving frame with a hybrid evolutionary algorithm (HEA) is proposed. In the first stage, a constructive heuristic is developed to determine the set of nozzle types assigned to each head and the total number of assembly cycles; in the second stage, constructive heuristics, an evolutionary algorithm with two evolutionary operators and a tabu search (TS) with multiple neighbourhoods are combined to solve all the sub-problems simultaneously, where the results obtained in the first stage are taken as constraints. Computational experiments show that the HEA can obtain good near-optimal solutions for small-size instances when compared with an optimal solver, Cplex, and can provide better results when compared with a TS and an EA for actual instances.

Keywords: Operation optimization; Mathematical model; hybrid evolutionary algorithm; PCB assembly.

1. Introduction

PCB assembly, through which the required electronic components are assembled onto PCBs, is very important to produce electronic products such as TVs and mobile phones etc. The most popular PCB assembly technique is surface mount technology (SMT), which involves three main operations including tin cream printing, component placement and joints welding. Only component placement is deemed as the "bottleneck" of the process because its operation machines, namely surface mounting machines (SMMs), are expensive and have low productivity (Ho et al., 2008). To remain competitive in the growing PCB market, PCB assembly companies are enthusiastic to improve the throughput rate of a PCB assembly line by minimizing component placement time taken by SMMs.

SMM usually has multiple heads, as shown in Figure 1. On this type of machine, the heads are sequentially arranged on a robot arm and the components are stored in stationary feeders. On other types of machine, heads may be arranged on a turret (Arob and Kendall, 2008). The issues are addressed to minimize component placement time involves (Ayob and Kendall, 2008, 2009): assigning nozzle types as well as components to heads, assigning feeders to slots and determining component picking-up and placement sequences. Most of the previous researches focus on one or two of the sub-problems. Until now, to our knowledge, no search has been done to consider all the sub-problems and solve them simultaneously.

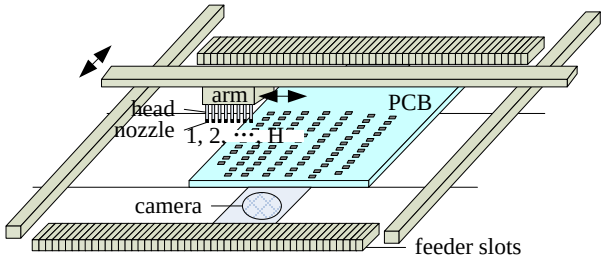

Figure 1 A SMM with an over gantry and multiple heads

Nozzle assignment is to assign nozzle types to heads to grasp different types of components, with the aim of minimizing the number of nozzle changes and balancing the workloads over heads. For the problem, constructive heuristics can be found in previous researches (Raduly-Baka et al., 2008; Guo et al., 2011; Knuutila et al., 2007; Ashayeri et al., 2011). Its mathematical models have been developed by Torabi et al. (2013) and Ashayeri et al. (2011) under different machine constraints.

Both feeder assignment and component placement sequencing problem have received great attentions in previous research. The two problems are often viewed quadratic assignment problem (QAP) and travelling salesman problem (TSP) or vehicle routing problem (VRP) (Ball and Magazine 1988, McGinnis et al. 1992; Or and Demirkol 1995, Jeevan et al., 2002, Grunow et al. 2004). The most popular constructive heuristic for the two sub-problems is nearest neighbour heuristic (Grunow et al., 2004, Pyottiala et al., 2013). Meta-heuristic approaches, especially evolutionary algorithms, have also been proposed to solve the two problems with different machine constraints, and some of them were combined with local searches. Ho and Ji (2003) developed a hybrid GA with 2-opt local search to solve the two sub-problems for a machine with a turret. This hybrid GA was then extended to solve the two sub-problems with different machine constraints (Ho and Ji, 2004, 2010, Ho et al., 2007, Ho et al. 2008). Duman and Or (2007) compared Taboo Search (TS), Simulated Annealing and GA for the two sub-problems. Chyu and Chang (2008) applied a GA-based algorithm with 2-opt local search to solve the two sub-problems on a turret style chip shooter machine. Alkaya and Duman (2015) combined simulated annealing and the heuristics for TSP to solve the two problems for a chip shooter component placement machines. As for the mathematical model, only few research results can be found. Altinkemer et al. (2000) provided an integrated mathematical model for the two sub-problems, Ho and Ji developed integer programming (IP) models for a kind of PAP machine (Ho and Ji, 2009) and a mathematical model for a turret machine (Ho and Ji, 2010), Luo and Liu (2014) proposed a MILP model to a SMM for LED assembly and Alkaya and Duman (2015) formulated the integration of the two problems to a nonlinear integer programming model.

It can be seen that less attention is paid on the integrated optimization of all the five sub-problems. In this regard, the integrated optimization problem of all the five sub-problems for a typical multi-head SMM is focused on in this paper. Specifically, a two-stage mathematical model and a two-stage solving frame are proposed for the problem. Constructive heuristics, an EA with two evolutionary operators and a tabu search (TS) with multiple neighbourhoods are combined to solve the sub-problems in two stages. The hybrid algorithm is compared with an optimizer Cplex, a TS and an EA. The comparison results indicate the efficiency of the proposed solving approach.

The remainder of this paper is organized as follows. A brief description of the concerned machine is provided in Section 2. Section 3 and Section 4 present a two-stage integer programming and a two-stage problem solving approach for the integrated operation optimization respectively. Computational experiments are reported in Section 5. Finally, conclusions are drawn in Section 6.

2. The Component Placement Process

The structure of the concerned machine is shown in Figure 1. Suppose that the feeders holding the required components have been set on slots, the required nozzles have been equipped on heads. The machine works with the following principles:

1) The PCB to be assembled is loaded to and fixed on the PCB platform.
2) The robot arm moves to the feeders to pick up H components in a certain pick-up order, then moves past a camera to detect and adjust components' postures.
3) The robot arm moves to the PCB and places the carried components to their corresponding placement locations in a certain placement order.
4) If there are still components to be placed, let the robot arm change the nozzle types on some heads if it is needed according to nozzle assignment, then repeat the operators of placing components as described in 2) and 3).

To simplify description, let the round trip of the robot arm picking up and placing H components be called an assembly cycle. Obviously, the longer length the robot arm travels to assembly components, the longer the placement time will take, so it is reasonable to minimize the total travelling distance of the robot arm in order to reduce component placement time. Obviously, the travelling distance is mainly determined by feeders' locations and the sequences of picking-up and placing components. It is also influenced by the assignments of nozzle types and components to heads since the components with a specific type can only be carried by the heads with a certain nozzle type and there is a fixed spacing distance (Δh) between two adjacent heads.

Because the robot arm moves simultaneously in the vertical and horizontal axes, let all the positions of placement locations and feeders be measured by Chebychev metric in a Cartesian coordinate system with x-y axis. The travelling distance of the robot arm is

composed of d_{ijhl}^{pp}, $d_{ijhls_i}^{fp}$, $d_{ijhls_is_j}^{ff}$, $d_{ijhls_j}^{pf}$, $d_{ihs_i}^{f}$ and d_{ih}^{p}, denoting the path distances the robot arm travels from placement location i to j, from feeder s_i to placement location j, from feeder s_i to s_j, from placement location i to feeder s_j, from location o to feeder s_i and from component placement location i to location o respectively, assuming that component i is carried by h and j by l and that the robot arm starts and ends up at a fixed location o.

The following assumptions are specified by the concerned multi-head SMM when the integrated optimization of the five sub-problems is considered:

1) One feeder can hold components for a single type.
2) One type of components can only be grasped by a nozzle with a certain type.
3) The set of heads is given by $H=\{H_m, H_s\}$ where H_m includes the moveable heads that can change their nozzles during placement process and H_s includes the static heads that cannot change their nozzles.
4) The set of nozzle types is given by $R=\{R_m, R_s\}$ where R_m includes the moveable nozzle types that can be changed by a moveable head during placement process and R_s includes the static heads that cannot be changed.

3. Modelling the optimization problem

3.1 Discussions on the problem features

Among the five sub-problems, the assignments of nozzles and components to heads are usually considered separately to balance the workloads over heads or to minimize the number of nozzle changes (Guo et al, 2011, Ashayeri et al. 2011, Knuutila et al, 2013), which can indirectly reduce the total placement time. However, their solutions only provide the number of components (workload) and the set of nozzle types assigned to each head, not the details of nozzle types and components in each assembly cycle. We follow this idea and deal with the two assignments in the first stage, and then reconsider them again in the second stage together with the other three sub-problems.

3.2 The MILP models

3.2.1 An MILP model for the assignments of nozzles and components to heads in the first stage

Based on the previous work on mathematical models (Tarobi et al., 2013 and Ashayeri et al., 2011), we develop a new mathematical model in which new constraints are added.

Parameters:

Q= the set of component types, indexed by $q \in Q$;

n_q = the number of components of type q;

b_{qr} = 1 if the components with type q can be grasped by nozzles with type r, 0 otherwise,

M = A very large number.

Variables:

$$u_{rh} = \begin{cases} 1, \text{if nozzle type } r \text{ is assigned to head } h, \\ 0, \text{otherwise.} \end{cases}$$

w_{qrh}= the number of the components with type q, grasped by nozzle type r on head h,

WL= the maximum workload on a head.

MILP model for the assignments of nozzles and components to heads:

$$\text{Minimize } \gamma_1 WL + \gamma_2 \sum_{h \in H} \sum_{r \in R} u_{rh} \quad (1)$$

St.

$$\sum_{q \in Q} \sum_{r \in R} w_{qrh} \le WL, \quad h \in H, \quad (2)$$

$$w_{qrh} \le u_{rh} b_{qr} n_q, \quad q \in Q, r \in R, h \in H, \quad (3)$$

$$\sum_{h \in H} \sum_{r \in R} w_{qrh} = n_q, \quad q \in Q, \quad (4)$$

$$\sum_{r \in R_m} u_{rh} \ge 1, \quad h \in H_m, \quad (5)$$

$$\sum_{r \in R} u_{rh} = 1, \quad h \in H_s, \quad (6)$$

$$\sum_{h \in H} u_{rh} \ge 1, \quad r \in R_m, \quad (7)$$

$$\sum_{h \in H_s} u_{rh} = 1, \quad r \in R_s, \quad (8)$$

$$u_{rh} = 0, \quad r \in R_s, \quad h \in H_m, \quad (9)$$

$$WL \in Z^+, w_{qrh} \in Z^+, u_{rh} = \{0, 1\}, \quad q \in Q, r \in R, h \in H. \quad (10)$$

The objective is to minimize the weighted sum of the maximum workload and the number of nozzle changes, where parameters γ_1 and γ_2 are the weighted coefficients. Constraints (2) express the maximum workload WL, larger than the workload over each head. Constraints (3) ensure that variable w_{qhr} is upper limited by n_q if nozzle type r is assigned to head h, zero otherwise. Constraints (4) indicate that all the components must be assigned to heads. Constraints (5) show that each moveable head should be assigned with at least one

moveable nozzle type and constraints (6) show that each static head could only be assigned with one nozzle type. Constraints (7) ensure that each moveable nozzle type must be assigned to at least one head, and constraints (8) and (9) ensure each static nozzle type can only be assigned to a static head and cannot be assigned to a moveable head.

A solution to the model provides the set of the nozzle types and the number of components assigned to a head and the maximum workload *WL* over heads. To make full use of each head, it is reasonable to require the assembly be finished in *WL* assembly cycles, i.e., the number of assembly cycles $G=WL$.

3.2.2 An MILP model for all the five sub-problems in the second stage

Based on TSP and QAP, a new MILP model is proposed. Different from previous models (Ho and Ji, 2010; Kim and Park, 2004; Jeevan et al., 2002; Luo and Liu, 2014; Alkaya and Duman, 2015), more sub-problems are integrated and more real constraints are considered in our model: 1) Component pick-up sequence which is often omitted is considered here; 2) Components with specific types can only be picked up by the heads with certain nozzle types, instead of by all the heads; 3) A moveable nozzle type can be changed only one time on each moveable head during assembly process.

Parameters,

S= the set of feeder slots, indexed by $s \in S$,

a_{iq} = 1 if the type of component i is q, 0 otherwise;

$c_{ir} = \sum_q a_{iq} b_{qr}$, i.e. c_{ir}=1 if a nozzle of type r can grasp component i, 0 otherwise,

d_{ijhl}^{pp} , $d_{ijhls_i}^{fp}$, $d_{ijhls_i s_j}^{ff}$, $d_{ijhls_j}^{pf}$, $d_{ihs_i}^{f}$ and d_{ih}^{p} - path distances in the six cases described in Section 2.

Variables:

x_{igh}=1 if component i is assigned to head h in assembly cycle g, 0 otherwise;

y_{ghl}=1 if the component on head l is placed following the component on head h in assembly cycle g, 0 otherwise; indexes h and l are permitted to be zero.

z_{ghl}=1 if component on head l is picked up following the component on head h in assembly cycle g, 0 otherwise; indexes h and l are permitted to be zero.

e_{qs}=1 if the feeder for component type q is assigned to slot s, 0 otherwise;

v_{ghr}=1 if head h is assigned with nozzle type r in assembly cycle g;

δ_{ghr}=1 if head h changes its current nozzle type to type r in assembly cycle g, 0 otherwise;

f_{ij}^{pp} = the distance the robot arm travels from placement location i to j, 0 otherwise;

f_{ij}^{ff} = the distance the robot arm travels from feeder s_i to s_j, 0 otherwise;

f_g^{fp} = the distance the robot arm travels from the feeder to the first placement location in assembly cycle g;

f_g^{pf} =the distance the robot arm travels from PCB to the feeder to pick up components in assembly cycle g;

f_0=the distance the robot arm travels from the original position to the feeder to pick up the first component;

f_e=the distance the robot arm travels from the last placement location to the original position.

MILP model for feeder assignment, component assignment and component sequencing:

$$\text{Minimize } D = f_0 + \sum_{i=1}^{N}\sum_{j=1, j\neq i}^{N}(f_{ij}^{pp} + f_{ij}^{ff}) + \sum_{g=1}^{G} f_g^{fp} + \sum_{g=2}^{G} f_g^{pf} + f_e \quad (12)$$

St.
$$f_{ij}^{pp} \geq d_{ijhl}^{pp} + (x_{igh} + x_{jgl} + y_{ghl} - 3)M, \quad \forall i, j(j\neq i), g, h, l(l\neq h) \quad (13)$$

$$f_g^{fp} \geq d_{ijhls}^{fp} + (x_{igh} + x_{jgl} + z_{gh0} + y_{g0l} + \sum_{q\in Q} e_{qs}a_{iq} - 5)M, \quad \forall i, j, g, h, l(l\neq h), s \quad (14)$$

$$f_{ij}^{ff} \geq d_{ijhls_1s_2}^{ff} + (x_{igh} + x_{jgl} + z_{ghl} + \sum_{q\in Q} e_{qs_1}a_{iq} + \sum_{q\in Q} e_{qs_2}a_{jq} - 5)M, \forall i, j(j\neq i), g, h, l(l\neq h), s_1, s_2(s_2\neq s_1) \quad (15)$$

$$f_g^{pf} \geq d_{ijhls}^{f} + (x_{i,g-1,h} + x_{jgl} + y_{g-1,h,0} + z_{g0l} + \sum_{q\in Q} e_{qs}a_{jq} - 5)M, \forall i, j(j\neq i), g>1, h, l(l\neq h), s \quad (16)$$

$$f_0 \geq d_{ihs}^{f} + (x_{i1h} + z_{10h} + \sum_{q\in Q} e_{qs}a_{iq} - 3)M, \quad \forall i, h, s \quad (17)$$

$$f_e \geq d_{ih}^{p} + (x_{iGh} + y_{Gh0} - 2)M, \quad \forall i, h \quad (18)$$

$$\sum_{g=1}^{G}\sum_{h\in H} x_{igh} = 1, \quad \forall i \quad (19)$$

$$\sum_{i=1}^{N} x_{igh} \leq 1, \quad \forall g, h \quad (20)$$

$$\sum_{l=0, l\neq h}^{H} y_{ghl} = 1, \quad \forall g, h\in\{0\}\cup H \quad (21)$$

$$\sum_{h=0, h\neq l}^{H} y_{ghl} = 1, \quad \forall g, l\in\{0\}\cup H \quad (22)$$

$$p_{gh} - p_{gl} + Hy_{ghl} \leq H - 1, \quad \forall g, h, l \quad (23)$$

$$\sum_{h=0, h\neq l}^{H} z_{ghl} = 1, \quad \forall g, h\in\{0\}\cup H \quad (24)$$

$$\sum_{h=0, h\neq l}^{H} z_{ghl} = 1, \quad \forall g, l\in\{0\}\cup H \quad (25)$$

$$t_{gh} - t_{gl} + Ht_{ghl} \leq H - 1, \quad \forall g \quad (26)$$

$$\sum_{s\in S} e_{qs} = 1, \quad \forall q \tag{27}$$

$$\sum_{q\in Q} e_{qs} \le 1, \quad \forall s \tag{28}$$

$$x_{igh} \le \sum_{r=1}^{R} v_{ghr} c_{ir}\ , \quad \forall i, g, h \tag{29}$$

$$v_{1hr} \le u_{rh}\ , \forall r, h \tag{30}$$

$$\sum_{r=1}^{R} v_{ghr} = 1, \forall\ g, h \tag{31}$$

$$v_{ghr} = v_{1hr}\ , \forall g\in G\backslash\{1\}, h\in H_s, r \tag{32}$$

$$\delta_{1hr} = v_{1hr}\ , \ \forall h\in H_m, r\in R_m \tag{33}$$

$$\delta_{ghr} \ge v_{ghr} - v_{g-1,h,r}\ , \ \forall g\in G\backslash\{1\}, h\in H_m, r\in R_m \tag{34}$$

$$\sum_{g=1}^{G} \delta_{ghr} \le 1, \forall h\in H_m, r\in R_m \tag{35}$$

$$x_{igh}, e_{qs}, v_{ghr}, \delta_{ghr}=\{0,1\}, g\in G, h\in H, q\in Q, s\in S, r\in R \tag{36}$$

$$y_{ghl}, z_{ghl}=\{0,1\}, g\in G, h,l\in\{0\}\cup H \tag{37}$$

$$p_{gh}, p_{gl}, t_{gh}, t_{gl}, f_g^{pf} \ge 0, \ g\in G\backslash\{1\}, h\in H, l\in H \tag{38}$$

$$f_{ij}^{pp}\ , \ f_{ij}^{ff}\ , f_0, f_e, f_g^{fp}\ , p_{gi}, t_{gi} \ge 0, g\in G, i, j\in J. \tag{39}$$

Objective (12) is the total distance that the robot arm travels to assemble all the required components. Constraints (13-18) define the distances that the robot arm travels in the six cases described in Section 2. Constraints (19) and (20) show that each component must be assigned to a head in a certain assembly cycle and a head in each assembly cycle is assigned with at most one component. Constraints (21-23) and (24-26) describe a placement sequence and a pick-up sequence of the components. Constraints (27-28) ensure that a feeder must be assigned to a slot and one slot is at most for one feeder. Constraints (29) ensure that a component *i* can be assigned to head *h* in assembly cycle *g* only if head *h* is assigned with a nozzle type for component *i*. Constraints (30) require that a nozzle type *r* can be assigned to head *h* only if u_{rh}=1 (obtained by the first stage model). Constraints (31) ensure that only one nozzle type can be assigned to each head *h* in each cycle *g*. Constraints (32) indicate that the nozzle type on a static head cannot be changed. Constraints (33) record a nozzle change in the first cycle while constraints (34) record a nozzle change in other cycles. Constraints (35) ensure that a moveable nozzle type *r* can be changed one time on each moveable head *h*. Constraints (36-39) define the ranges of all the variables.

The objective function together with constraints (13-39) involves a TSP and an assignment problem, which are two well-known NP-hard problems. Therefore, it could be

solved to optimality only for very small-size of problem instances. Consequently, it is necessary to develop heuristic approach to solve large-size problem instances.

4. The proposed heuristic approach

A two-stage approach composed of EA with two evolutionary operators, TS with multiple neighbourhoods and several constructive heuristics is proposed to solve the considered problem. In the following, the way of representing a solution is explained firstly, followed by the analysis of the optimal pick sequence in an assembly cycle, and then the main components of the approach.

4.1 Solution representation

To simplify description, symbols H, S and Q are also used to denote the cardinalities of sets H, S and Q respectively without ambiguousness. The solution for each sub-problem is represented by cluster and permutation. A nozzle assignment is represented by $NA=\{N_1, \ldots N_g, \ldots N_G\}$ where $N_g=\{n_{g1}, \ldots n_{gh}, \ldots, n_{gH}\}$ and n_{gh} denotes the nozzle type on head h in assembly cycle g. Similarly, an assignment of components to heads is represented by $WA=(W_1, W_2, \ldots, W_G)$ where $W_g=\{w_{g1},\ldots w_{gh}, \ldots, w_{gH}\}$ and $w_{gh}\in\{0\}\cup J$ denotes the component assigned to head h. A pick-up sequence and a placement sequence of components in assembly cycle g are represented as $P_g=(p_{g1},\ldots, p_{gk}, \ldots,p_{gH})$ and $M_g=(m_{g1},\ldots, m_{gk}, \ldots, m_{gH})$ respectively where p_{gk} and m_{gk} denote the component that is picked up at k^{th} step and the component that is placed at k^{th} step respectively. To simplify description, let $PS=(P_1, \ldots, P_g, \ldots, P_G)$ and $MS=(M_1,\ldots, M_g, \ldots, M_G)$. A feeder assignment is represented as $E=\{e_1, e_2, \ldots, e_q, \ldots, e_Q\}$ where $e_q\in[1\ S]$ denotes the slot to which the feeder for component type q is assigned. Values of w_{gh}, p_{gk} and m_{gk} may be zero, denoting that no component is assigned to head h or no component is picked up or placed at $k^{\text{-th}}$ step in assembly cycle g.

4.2 The optimal pick sequence in a given assembly cycle

When other decisions are given, the optimal pick-up sequence of the components $\{w_{g1}, \ldots, w_{gl}, \ldots, w_{gH}\}$ in assembly cycle $g(1<g<G)$ must be one of the two orders that the components encounter their corresponding heads when the robot arm travels directly from one side to the other side along slots. Obviously, the arm must travel to a leftmost position

FL_g and a rightmost position FR_g to pick up all the components in assembly cycle g. The travelling distance that the arm moves directly from FL_g to FR_g (or from FL_g to FR_g) must be shorter than the one that the arm reciprocates forward, according to the triangle inequality. That is, each component should be picked up when it encounters its head when the arm travels directly from one side to the other along slots, which results in the minimal travelling distance of the arm. However, there are two such orders of the components and one order is the reverse of the other. The best order should be the one corresponding to the minimal traveling distance that the arm starts from the last placement location in assembly g−1, moves along the slots to pick up the components, and arrives to the first placement position in assembly g.

4.3 The outline of the two-stage solving approach based on EA and TS

Corresponding to the two-stage model, a two-stage solving approach is developed. In the first stage, the set of nozzle types assigned to each head and the total number of assembly cycles are to be determined by a constructive heuristic, while in the second stage, the sub-problems are solved by a hybrid algorithm (HEA) of EA and TS, taking the results obtained in the first stage as constraints. The flowchart of HEA is shown in Figure 2.

Algorithm HDDE:
Output: The best solution X_b=(NZ_b, W_b, E_b, PS_b, MS_b)
Stage 1:
1.1 Assign components and nozzle types to each head by using **heuristic 1** and output the set of nozzle types assigned to each head and the total number of assembly cycles G.
Stage 2:
2.1 Generate an initial population: generate feeder assignments by using **heuristic 2**, component assignments and nozzle assignments by using **heuristic 3**, pick-up sequences by using a **optimal rule** and placement sequences by using **heuristic 4**. The initial population is denoted as P_0= $\{X_1^0,...,X_i^0,...,X_{NP}^0\}$ where $X_i^0=(NZ_i^0,WA_i^0,E_i^0,PS_i^0,MS_i^0)$
2.2 Repeat
For (i=1,2,⋯, NP)
2.2.1 Improve population $\{WA_1^t,...,WA_i^t,...,WA_{NP}^t\}$ by using ***GA*** operator;
2.2.2 Improve population $\{E_1^t,...,E_i^t,....E_{NP}^t\}$ by using ***DDE*** operator;
2.2.3 Improve population $\{E_1^t,...,E_i^t,....E_{NP}^t\}$ and $\{WA_1^t,...,WA_i^t,...,WA_{NP}^t\}$ by using **TS** with multiple neighborhoods.
2.2.4 Update the best solution X_b,
Until (the iteration number t≥MaxIter or the continuous non-improvement iteration number nt≥MaxNIter)

Note: PS_i^t and MS_i^t are regenerated once component assignment WA_i^t is changed; PS_i^t is regenerated if feeder assignment E_i^t is changed; otherwise $PS_i^t=PS_i^{t-1}$, $WA_i^t=WA_i^{t-1}$.

Figure 2 The frame of the proposed two stage approach HEA

In HEA, an EA algorithm with two evolutionary operators (denoted as GA and DDE in Figure 2) is developed to evolve a feeder assignment and a component assignment. A TS with multiple neighbourhoods is developed to further improve solutions. The pick-up and the placement sequences for the components in each assembly cycle are generated by using the optimal rule described in Section 4.2 and a constructive heuristic respectively when a feeder assignment and a component assignment are given.

4.3.1 Heuristic for the assignments of nozzle types and components in the first stage (Heuristic 1)

The aim of assigning nozzle types and components to heads in the stage is to minimize the maximum workload over heads and the number of nozzle changes. Notice that the maximum workload is the minimum if all the heads are assigned with the same number of components, and the number of the nozzle changes is small when a nozzle type is permitted to be changed one time on a moveable head. The following heuristic tries to reduce the maximum workload over heads.

Step 1: Assign an unassigned static nozzle type to an arbitrary unoccupied static head until all the static nozzle types are assigned to heads.

Step 2: Assign an unassigned moveable nozzle type to an arbitrary moveable head until all the moveable nozzle type are assigned with heads or no moveable head is unoccupied.

Step 3: Assign the moveable nozzle type with the maximum workload to an arbitrary unoccupied moveable head until all the moveable heads are assigned with nozzle types.

Step 4: Take all the moveable nozzle types as a union nozzle type. Assign the nozzle type with the maximum average workload of head to an arbitrary unoccupied static head until all the static heads are assigned with nozzle types.

The heuristic determines the set of nozzle types assigned to each head h, denoted as U_h, and the maximum workload over heads, i.e., the number of assembly cycles. According to the maximum number of basic operations in each step, the complexity of the heuristic is $O(HQ)$ if $Q \geq R$ holds or $O(HR)$ otherwise.

4.3.2 The initial feeder assignment solutions (Heuristic 2)

NNH and random assignment are two common ideas to generate a feeder assignment in previous researches (Pyottiala et al, 2013; Ho and Ji, 2010; Ho et al., 2007; Kulak et al., 2007). We follow the two ideas and assign a feeder to a slot in one of the two following ways, each with 50% probability: 1) assign a feeder to an arbitrary slot; 2) assign a feeder to an arbitrary slot in the set of Q slots near to the centre of the PCB. Repeat the above procedure NP times, a population with NP feeder assignment solutions is generated.

4.3.3 The initial component assignment and nozzle assignment in the second stage (Heuristic 3)

The set of components in each assembly cycle limits the components involved in the placement sequence for this assembly cycle, so it is reasonable to take the distances among placement locations into account when component assignment solution is generated. The NNH based heuristic to generate a component assignment WA and a nozzle type assignment NA is as follows.

Step 1: Let Ω_r ($r \in R$) include the components that could be grasped by nozzle type r. Set assembly cycle number g=1.

Step 2: For h=1 to H: assign an arbitrary nozzle type $r' \in U_h$ to head h, i.e., set $n_{gh}=r'$. Move r' from U_h.

Step 3: Form a component set for assembly cycle g:

3.1 If no unassigned component can be carried by head h, i.e., $\Omega_{r(h)}=\varnothing$, go to Step 3.3; otherwise, assign an arbitrarily component $i \in \Omega_{r(h)}$ to head h, i.e., set $w_{gh}=i$; set $\Omega_{r(h)}= \Omega_{r(h)} \backslash \{i\}$, $h=h+1$ and record the latest assigned component $w_{g0}=i$.

3.2 If no unassigned component can be carried by head h, i.e., $\Omega_{r(h)}=\varnothing$, go to Step 3.3; otherwise, assign the nearest component $i \in \Omega_{r(h)}$ to the placement location of w_{g0} to head h, i.e., set $w_{gh}=i$; set $\Omega_{r(h)}= \Omega_{r(h)} \backslash \{i\}$, $h=h+1$ and $w_{g0}=i$; if $h>H$, go to Step 4, otherwise repeat this step.

3.3 If head h is a moveable head, change the current nozzle type to an arbitrary type $r' \in U_h$, i.e., set $n_{gh}=r'$, and move it from U_h, i.e., set $U_h=U_h \backslash \{r'\}$. If head h is a static head, set $w_{gh}=0$ and $h=h+1$. Go to Step 4 if $h>H$, return to Step 3.1 if both $w_{g0}=\varnothing$ and $h \le H$ hold, go to Step 3.2 otherwise.

Step 4: Set $g=g+1$. If $g \le G$ holds, return to Step 3.1 after setting $NA_g=NA_{g-1}$ and h=1; otherwise terminate the heuristic.

In the heuristic, the distance of two placement locations is measured by the distance the robot arm travels between them instead of their Chebyshev distance. The maximum number of basic operations exists in Step 3.2. In the worst case, the complexity of the above heuristic is O(*GN*).

4.3.4 Generating component placement sequence (Heuristic 4)

Determining the optimal placement sequence of the components in each assembly cycle is equivalent to solving a TSP problem. Inspired by this, a constructive heuristic based on NNH is developed to generate a component placement sequence in each assembly cycle.

Step 1: Identify two extreme placement locations F_l and F_r with the minimum and maximum x-coordinate values respectively in assembly cycle g. Schedule the component corresponding to location F_l as the first one in the sequence. Set sequence index k=2.

Step 2: Schedule the component nearest to the last sequenced component as the k-th in the sequence and set k=k+1. Repeat this step until the component for location F_r is scheduled, resulting in a partial component placement sequence.

Step 3: Schedule each un-scheduled component one by one at its best position in the partial placement sequence such that the robot arm travels the minimum distance when it follows the obtained placement sequence.

The distance of two placement locations is also measured by the distance the robot arm travels instead of their Chebyshev distance. Basically, the above procedures follow those of NNH. So the complexity of the above heuristic is $O(H)$, the same as that of NNH for H cities.

4.3.5 An EA with two evolutionary operators

Evolutionary operator is the most important component of an EA. Many evolutionary operators such as partially mapped crossover, cycle crossover (CX), exchange mutation etc. have been used in GAs to evolve a solution (Ho and Ji, 2010; Hardas et al., 2008; Li et al., 2008; Chen and Lin, 2007; Ho et al., 2007; Kulak et al., 2007). Different from these operators, DDE operator is to generate a trial individual by adding the weighted difference between two target individuals in the population at the previous iteration to the third target individual. An EA with two evolutionary operators are designed as follows.

1) GA with cycle crossover (CX) for component assignment

CX operator (Kulak et al., 2007, Burke and Kendall, 2005) can always result in feasible offspring. A CX is developed here according to the features of component assignment.

(1) For each individual W_i^t in population at generation *t*, if a random number $r \in U(0, 1)$ is less than P_{cr}, execute the following cycle crossover, otherwise set $W_i^{t+1} = W_i^t$.

(2) Select a solution W_x^t by using wheel rotation method, sort all the components of W_x^t in a sequence on head and cycle numbers, and then execute the cycle crossover operator as shown in Figure 3. Retain the best one of $\{W_i^t, O_{i1}^t, O_{i2}^t\}$ as W_i^{t+1}.

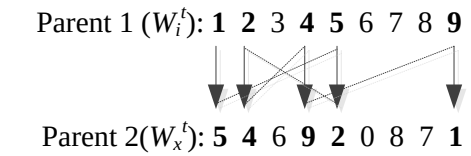

Figure 3 The cycle crossover operator

(3) Mutation: if a random number $r \in U(0, 1)$ is less than P_m, randomly swap two components with the same nozzle types in W_i^{t+1}.

2) DDE with DE/rand/1 operator for feeder assignment

DE/rand/1 mutation (Das and Suganthan, 2011; Neri and Tirronen, 2010) which has been indicated to be one of the most efficient operators is adopted here.

(1) For three feeder assignment solutions E_u^t, E_r^t and E_x^t at generation *t*, execute the following mutation procedures:

$$V_{ij}^t = L(E_{uj}^t + E_{rj}^t - E_{xj}^t)$$

where V_{ij}^t is the element generated for the *j*-dimension of the *i*-th solution at generation *t*, and function $L(x)$ is a mapping: $x \rightarrow S$ (the set of slots) that $L(x)=x$ if $1 \le x \le |S|$, $L(x)=|S|+x$ if $x \le 0$ and $L(x)=x-|S|$ if $x>|S|$. Let $E_u^t = E_i^t$, $E_r^t = E_b$ and E_x^t be an arbitrary feeder assignment at generation *t* in our implementation to give fair chance to each solution and preference to the current best solution E_b.

(2) Following the mutation phase, the trial individual is obtained by using crossover operator such that:

$$U_{ij}^{t}=\begin{cases}V_{ij}^{t}, \text{if } r_2 \le P_c \text{ and slot } V_{ij}^{t} \text{ is available,}\\ E_{ij}^{t}, \text{if } r_2 > P_c \text{ and slot } E_{ij}^{t} \text{ is available,}\\ s, \text{otherwise.}\end{cases}$$

where r_2~U[0,1], P_c is the crossover probability and s is an arbitrary unoccupied slot.

(3) Selection: If U_i^t is better than E_i^t, let $E_i^t = U_i^t$.

4.3.6 A simple TS method with multiple neighbourhoods

TS is a widely used algorithm for NP-hard optimization problems (Glover, 1989, 1990) because of its simplicity and efficiency. A TS is developed here to improve a solution. Local search and tabu list is the key components in a TS.

1) Local search adopted in TS

Local search is the iteration of neighbourhood searches. In previous researches (Ho and Ji, 2010; Chyu and Chang, 2008; Ho et al., 2007; Kulak et al., 2007; Grunow et al., 2004; Duman and Or, 2007), 2-opt local search and iterated swap procedures (ISP) have been adopted to improve a solution for feeder assignment or component sequencing problem. In our paper, several neighbourhoods are defined to develop efficient local searches.

N_{shift}^{E} ={ $E' \mid E'$ is obtained from E by shifting a feeder from its current slot to an unoccupied slot};

$N_{exchange}^{E}$ ={$E' \mid E'$ is obtained from E by exchanging the slots of two feeders}.

N_{shift}^{G-in} ={ $W' \mid W'$ is obtained from W by shifting a component from one head to an unoccupied head};

$N_{exchange}^{G-in}$ ={ $W' \mid W'$ is obtained from W by exchanging the components on two heads};

N_{shift}^{G-ex} ={ $W' \mid W'$ is obtained from W by shifting a component subset for a mount cycle to another position in the sequence of all the component subsets in W};

$N_{exchange}^{G-ex}$ ={ $W' \mid W'$ is obtained from W by exchanging the positions of two component subsets}.

$N_{exchange}^{G-in}$ has the largest size of at most $N(N-1)/2$. Although it is not too large, the evaluating of a neighbourhood move is time-consuming because the component pick-up and placement sequences in an assembly cycle need to be regenerated once the components involved in the assembly cycle are changed. To speed up the computation, restriction technique is used as follows to tune neighbourhood $N_{exchange}^{G-in}$ to smaller, less powerful but easier to evaluate the neighbourhood move: 1) the exchange of components i and j is permitted only if the placement location of component i is near that of component j, i.e., the horizontal difference d_{ij}^{x} or the vertical coordinate difference d_{ij}^{y} between the two placement locations satisfies $d_{ij}^{x} \le \Delta h \times n_1$ or $d_{ij}^{y} \le \Delta h \times n_2$, where Δh is the spacing distance between two adjacent heads, $n_1=H/2$ and $n_2=2$ are predetermined parameters; 2) the first improvement rule is used to select a neighbour of a solution, i.e., we accept the first solution encountered with better cost.

2) Tabu list

Tabu list is another key strategy of TS, used to avoid repeated searches. In this paper, we memorize the component and the assembly cycle involved in the current performed move in the tabu list. Any neighbourhood move in the local searches of $N_{exchange}^{G-in}$ and N_{shift}^{G-ex} that involves a component or an assembly cycle in the tabu list is forbidden to perform in the next TL generations except when yields better solution.

In the procedures of TS, although not explicitly described, the component pick-up and placement sequences in an assembly cycle are always regenerated to identify an improved neighbour once the participated components in the cycle are changed. Considering that it is time-consuming to execute TS at each generation, a probability (P_s) is set to select a solution to which the TS is applied. The value of P_s is determined by pre-experiments that are described in Section 5.1.

5. Computational experiments

The HEA and the compared algorithms were implemented in C++ and run on a 2.4GHz computer with 2.0 GB Ram. The multi-head SMM used in the experiments has the following settings: 5 nozzle types, 8 heads and 50 slots on one side, where nozzle types (1 2, 3) can be changed by moveable heads (2, 4, 6, 8); the distance between two adjacent heads is 21mm.

The production data for 20 different PCBs were collected from practical production to evaluate the proposed heuristic. The sizes of the instances are shown in Table 1.

Table 1 The sizes of tested instances.

No.	N	Q	No.	N	Q
1	8	3	11	49	19
2	16	14	12	50	20
3	17	9	13	52	6
4	19	9	14	54	22
5	21	4	15	78	33
6	21	16	16	95	12
7	24	5	17	117	20
8	25	9	18	168	48
9	24	4	19	306	14
10	32	3	20	378	12

5.1 Setting the model and algorithm parameters

Model parameters γ_1 and γ_2 weigh the importance of the two objectives in formula (1). In the experiments, we set γ_1=0.6 and γ_2=0.4 to give more emphasis on minimizing workload than minimizing the number of nozzle changes. However, it could be any other weight pairs, decided by decision maker.

For algorithm parameters, length of tabu list *TL* is set to a representative value 9, and crossover probability in DDE operator is set as P_c=1−5/Q, where P_c varying with Q tries to give more chances to the HEA to explore its search for the instances with large number of component types. Parameters P_{cr}, P_m, NP and P_s are determined by using orthogonal experiments. The levels of the parameters are given in Table 2. MaxIter and MaxNIter are set according to experiments.

Table 2 Combinations of algorithm parameters in experiments

Levels	P_{cr1}	P_{m1}	NP	P_s
1	0.6	0.1	5	0.1
2	0.7	0.2	10	0.2
3	0.8	0.3	20	0.3
4	0.9	0.4	30	0.4

The orthogonal experiment is carried out on the instance (No.20) with the largest size, which is the most complex instance we collected. The computational time is limited to 100s and no limit is set for parameters *MaxIter* and *MaxNiter*. An orthogonal array $L_{16}(4^4)$ is selected. For each parameter combination, the proposed algorithm is run independently 20

times. The algorithm performance under each level of each parameter can be calculated. Figure 4 reports the factor level trend of the four parameters.

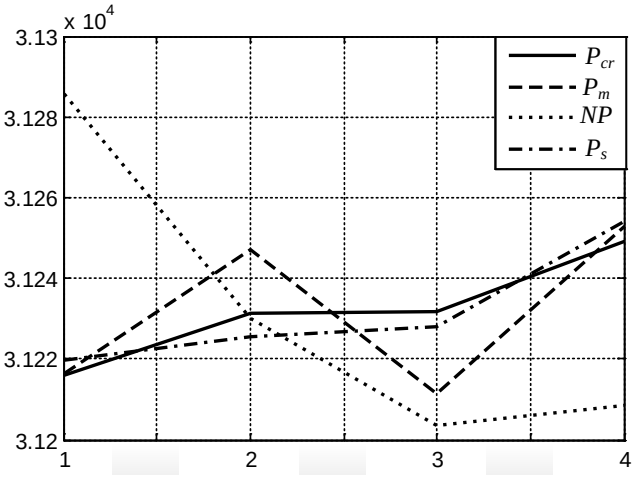

Figure 4 Factor level trends of four key parameters in terms of average objective function values

It can be seen from Figure 4 that the proposed algorithm has better performance under the four parameters with the following values: P_{cr}=0.6 (level 1), P_m=0.3 (level 3), NP=20 (level 3) and P_s=0.1 (level 1). We can also see that NP are more critical than other parameters according to the values of the standard deviations.

Under these parameter settings, the algorithm is run 20 times for instance No.20 again. We found that the algorithm terminates less than 150 generations and no better solution could be found when 60 continuous un-improvement generations is exceeded, so we set the permitted maximum generation MaxIter=150 and the permitted continuous un-improvement generation number MaxNiter=60.

5.2 Performance analysis of each component in HEA

To investigate the effectiveness of each component of HEA, we compare HEA with the variant HEA_{-TS}, HEA_{-GA} and HEA_{-DDE}, in which TS, GA and DDE are taken out one by one. To compare the algorithms, relative percentage increase (*PRI*) is calculated as:

$$PRI = (D_c - D_b) / D_b$$

where D_b is the objective function value of the best solution found by all the compared algorithms, while D_c is the objective function value of the best solution generated by a given algorithm. The PRI values in terms of average objective function value are shown in Table 3.

Table 3 Comparison of PRI values for four algorithms

Instance	HEA	HEA-GA	HEA-DDE	HEA-TS
1	**0.00**	**0.00**	**0.00**	15.20
2	**0.00**	0.37	0.38	24.81
3	**0.00**	0.51	0.33	8.87
4	**0.00**	2.74	1.23	6.85
5	**0.00**	0.45	0.76	10.04
6	**0.00**	0.23	0.43	19.43
7	**0.00**	0.46	1.42	10.52
8	0.52	1.43	**0.00**	10.24
9	0.08	**0.00**	0.06	11.15
10	0.23	1.37	**0.00**	9.87
11	0.64	1.63	**0.00**	20.89
12	0.01	0.83	**0.00**	19.93
13	0.52	0.69	**0.00**	9.05
14	0.19	1.10	**0.00**	23.12
15	**0.00**	1.95	0.11	38.76
16	0.23	0.55	0.00	19.77
17	0.18	1.21	**0.00**	24.06
18	**0.00**	1.56	0.43	41.76
19	**0.00**	1.01	0.02	43.33
20	0.14	0.75	**0.00**	35.59
Average	0.14	0.94	0.26	20.16
Deviation	0.20	0.69	0.42	11.63

It can be observed that HEA obtains better values for all the given 20 instances than TEA_{-TS} and TS is critical to the HEA. It can also be seen that there is very small difference between the HEA, HEA_{-GA} and HEA_{-DDE} according to PRI values. GA and DDE operators only have small contributions to the algorithm performance of HEA. But when taking both GA and DDE out, the algorithm converges very slowly according our experiments.

5.3 Comparison of the heuristic results with the optimal solutions for small-size instances

The HEA is executed on small-size instances and the results are compared with those obtained by a MILP optimizer, Cplex. Fifteen small-size instances with 10 slots are generated by simulating the real production situation. The optimizer is permitted to run in 600s while HEA runs 20 times for each instance and the best solution is collected. The obtained results are included in Table 4. It can be seen that the results of HEA are near to the optima as the average gap is 5.04% and HEA can solve all the instances in short length of running time. The reason that HEA does not obtain the optima may be that several constructive heuristics are included.

Table 4 Comparison of the heuristic results with the optimal solutions

Scale	No.	Objective function values (mm)		$\frac{(D_{HEA}-D_{cplex})}{D_{cplex}}\times 100\%$	Computation time (s)	
$N\times H$		D_{HEA}	D_{cplex}		t_{HEA}	t_{cplex}
3×3	1	663.10	640.31	6.06	0.07	0.22
	2	457.12	424.66	11.24	0.07	0.27
	3	592.40	548.78	5.03	0.07	0.39
	4	649.22	637.76	6.82	0.07	0.97
	5	567.32	522.41	8.60	0.07	0.27
4×4	1	700.72	668.72	4.79	0.08	14.98
	2	469.09	453.09	4.26	0.07	463.03
	3	592.45	576.40	5.56	0.07	33.49
	4	667.94	640.48	6.02	0.11	285.40
	5	583.32	552.86	5.51	0.07	19.08
8×8	1	801.49	unsolved	-	0.08	-
	2	520.32	unsolved	-	0.08	-
	3	658.45	unsolved	-	0.08	-
	4	733.01	unsolved	-	0.09	-
	5	633.63	unsolved	-	0.09	-
Average				5.04	0.08	

5.4 Comparison of HEA with TS, EA and the constructive heuristic in practical system

Experiments on a TS and a EA are carried out to evaluate the performance of HEA. The TS begins with a random initial solution, executes the same improvement procedures in Section 3.3.6. The EA is a variant of HEA, taking TS out from HEA. Within the maximum running time taken by HEA, both of the two algorithms run 20 times for each instance, and the obtained results are shown in Table 5. The objectives of the solutions provided by the constructive heuristic in the current production systems are also listed in column H_{ori}. To illustrate the convergence of the compared algorithms, the curves of the objective values varying with time for a relatively large instance No.20. are also shown in Figure 5.

There are few observations we would like to highlight as follows according to Table 5:

(1) HEA outperforms TS and EA in terms of minimum objective function values for most instances, and obtained the minimum average PRI value. So, it can be seen that HEA is the most efficient.

(2) In terms of the average objective function values, HEA also obtains the minimum average PRI value and performs best. It can be concluded that HEA is the most robust. Besides, from Figure 5, it can be seen that HEA has the best convergence property and can easily escape from local optima.

(3) It can be seen that the solutions obtained by HEA, TS and EA for each instance make improvements on the original ones provided by the current system (as shown in column H_{ori}). This is mainly because the current system is limited to constructive heuristic only.

Table 5 Comparisons of the PRI values of HEA, TS and EA

Instance	Minimal Objective function values			Average objective function values			Objective function values
	HEA	TS	EA	HEA	TS	EA	H_{ori}
1	0.00	0.60	14.95	0.00	0.00	7.42	176.49
2	0.00	7.92	26.36	0.00	1.78	13.86	128.82
3	0.00	2.39	8.88	0.00	2.61	7.42	53.89
4	0.00	5.67	5.91	0.00	2.88	4.05	20.23
5	0.00	4.78	7.56	0.00	0.89	6.53	16.81
6	0.00	9.61	14.75	0.00	3.25	9.19	56.93
7	0.00	7.72	9.87	0.00	2.63	5.88	543.34
8	0.00	4.74	8.62	0.00	2.18	5.02	170.95
9	0.00	8.08	9.79	0.00	4.53	4.02	142.60
10	0.00	4.05	9.16	0.00	1.86	7.73	50.69
11	0.00	4.28	12.57	0.00	1.63	6.45	63.06
12	0.00	3.64	9.41	0.00	1.30	4.45	67.13
13	0.00	3.37	4.23	0.00	1.85	2.41	97.06
14	0.00	3.97	10.25	0.00	2.36	6.50	75.87
15	0.00	6.42	18.21	0.00	3.21	7.70	89.22
16	0.00	3.70	8.54	0.00	2.96	7.59	100.93
17	0.00	3.91	10.44	0.00	2.49	8.10	84.71
18	0.00	5.41	13.50	0.00	5.59	12.46	55.14
19	0.00	2.22	13.67	0.00	1.75	11.85	80.20
20	0.00	2.14	14.93	0.00	1.92	10.71	67.10
Average	0.00	4.73	11.58	0.00	2.38	7.47	107.06

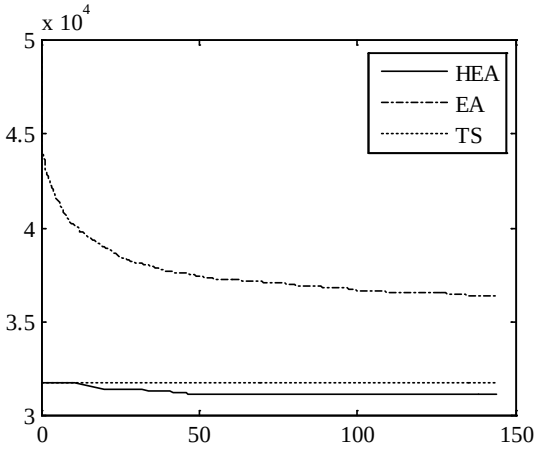

Figure 5 Curves of the objective function values varying with time for instance No. 20.

6. Conclusions

This paper considers a complex optimization problem arising in PCB assembly, which includes five interrelated sub-problems. To solve it efficiently, a two-stage mathematical programming model and a two-stage hybrid evolutionary algorithm are proposed. In the first stage, a constructive heuristic is developed to determine the set of nozzle types assigned to each head and the total number of assembly cycles; in the second stage, constructive heuristics, an evolutionary algorithm (EA) with two evolutionary operators and a tabu search (TS) with multiple neighbourhoods are combined to solve all the sub-problems, taking the results obtained in the first stage as constraints. Experimental results show that HEA has better performance than other compared algorithms. In future, we plan to investigate the integration optimization problem of PCB assembly which involves several surface mounting machines.

Acknowledgements

The work is supported by the Fundamental Research Funds for the Central Universities of China (Grant No. 2014z0033).

References

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

Altinkemer, K., B. Kazaz, M. Köksalan and H. Moskowitz. Optimization of printed circuit board manufacturing: Integrated modelling and algorithms. European Journal of Operational Research, 2000, 124: 409-421.

Ashayeri, J., N. Ma and R. Sotirov. An aggregated optimization model for multi-head SMD placements. Computers & Industrial Engineering, 2011, 60: 99-105.

Ayob, M., and Kendall, G. A survey of surface mount device placement machine optimization: Machine classification. European Journal of Operational Research, 2008, 186(3): 896-914.

Ayob, M., and Kendall, G. The optimization of the single surface mount device placement machine in printed circuit board assembly: a survey. International Journal of Systems Science, 2009, 40(6): 553-569.

Ball, M. and M. Magazine. Sequencing of insertions in printed circuit board assembly. Operations Research, 1988, 36: 192–201.

Burke E, and Kendall G. Search methodologies, introductory tutorials in optimization and decision support techniques. Springer, 2005.

Chen, Y.M. and C.T. Lin. A particle swarm optimization approach to optimize approach placement in printed circuit board assembly. International Journal of Advanced Manufacturing Technology, 2007, 35(1-8): 610-620.

Chyu, C.C., and Chang, W.S. A genetic-based algorithm for the operating sequence of a high speed chip placement machine. International Journal of Advance Manufacturing Technology, 2008, 36(9-10): 918-926.

Das, S., and Suganthan, P.N. Differential Evolution: A survey of the state-of-the-art. IEEE Transactions on Evolutionary Computation, 2011, 15(1): 4-31.

Duman, E. and Or, I. The quadratic assignment problem in the context of the printed circuit board assembly process. Computers & Operations Research, 2007, 34(1): 163-179.

Glover, F. Tabu Search - Part 1. ORSA Journal on Computing 1 (2): 190–206, 1989.

Glover, F. Tabu Search - Part 2. ORSA Journal on Computing 2 (1): 4–32, 1990.

Grunow, M., Gunther, H., Schleusener, M., and Yimaz, I.O. Operations Planning for collect-and-place machines in PCB assembly. Computers & Industrial Engineering, 2004, 47(4): 409-429.

Guo, S.J, Takahashi, K and Morikawa, K. PCB assembly scheduling with alternative nozzle types for one component type. Flexible Services and Manufacturing Journal, 2011, 23(3): 316-345.

Hardas, C.S., Chinmaya, S. Doolen, T.L. and Jensen, D.H. Development of a genetic algorithm for component placement sequence optimization in printed circuit board assembly. Computers & Industrial Engineering, 2008, 55(1): 165-182.

Ho, W. and P. Ji. Component scheduling for chip shooter machines: a hybrid genetic algorithm approach. Computers & Operational Research, 2003, 30: 2175-2189.

Ho, W., and Ji, P. A hybrid genetic algorithm for component sequencing and feeder arrangement. Journal of Intelligent Manufacturing, 2004, 15(3): 307-315.

Ho, W. and P. Ji. Integrated component scheduling models for chip shooter machines. International Journal of Production Economics, 2010, 123: 31-41.

Ho, W., P. Ji and P.K. Dey. Optimization of PCB component placements for the collect-and-place machines. International Journal of Advanced Manufacturing Technology, 2008, 37(7-8): 828-836.

Ho W, Ji P and Wu Y. A heuristic approach for component scheduling on a high-speed PCB assembly machine. Production Planning & Control, 2007, 18(8): 655-665.

Jeevan, K., A. Parthiban, K. N. Seetharamu, I. A. Azid, and G. A. Quadir. Optimization of PCB component placement using genetic algorithms. Journal of Electronics Manufacturing, 2002, 11 (1): 69 – 79.

Kim, K.M., and Park, T.H. PCB assembly optimization of chip mounters for multiple feeder assignmen. SCIE 2004 Annual Conference, 1(3): 1425-1430.

Knuutila, T., Pyottiala and Nevalainen, O.S. Minimizing the number of pickups on a multi-head placement machine. Journal of the Operational Research Society, 2007, 58(1): 115-121.

Knuutila, T., Suomi, T., Emet, S., Johnsson, M., and Nevalainen, O.S. Organizing the nozzle magazine of a gantry-type PCB assembly machine. International Journal of Advanced Manufacturing Technology, 2013, 68(5-8): 1189-1202.

Kulak, O., Yilmaz, I.O., and Guenther, H.O. PCB assembly scheduling for collet-and-place machines using genetic algorithms. International Journal of Production Research, 2007, 45(17): 3949-3969.

Li, S.Y., Hu, C.F., Tian, F.H. Enhancing optimal feeder assignment of the multi-head surface mounting machine using genetic algorithms. Applied Soft Computing, 2008, 8(1): 522-529.

Luo, J. and J. Liu. An MILP model and clustering heuristics for LED assembly optimisation on high-speed hybrid pick-and-place machines, International Journal of Production Research, 2014, 52(4): 1016-1031.

McGinnis, L. F., J. C. Ammons, M. Carlyle, L. Cranmer, G. W. DePuy, K. P. Ellis, C. A. Tovey, and H. Xu. Automated Process Planning for Printed Circuit Card Assembly. IIE Transactions, 1992, 24 (4): 18 – 30.

Neri, F., Tirronen, V. Recent advances in differential evolution: a survey and experimental analysis. Artificial Intelligence Review, 2010, 33: 61-106.

Or, I., and Demirkol, E. Optimization issues in automated production of printed circuit boards: Operations sequencing and feeder configuration problems. In Proceedings of 1995 INRIA/IEEE Symposium on Emerging Technologies and Factory Automation, Paris, France, October 10-1, 1995.

Raduly-Baka, C., Knuutila, T., Johnsson, M., and Nevalainen, O.S. Selecting the nozzle assortment for a Gantry-type placement machine. OR Spectrum, 2008, 30: 493-513.

Pyottiala, S., Knuutila, T., Johnsson, M., Nevalainen, O.S. Minimizing the assembly cycle time on a revolver gantry machine. Computers & Operations Research, 2013, 40(11): 2611-2624.

Tarobi, S.A., Hamedi, M., and Ashayeri, J. A new optimization approach for nozzle selection and component allocation in multi-head beam-type SMD placement machines. Journal of Manufacturing Systems. 2013, 32(4): 700-714.