An integrated scheduling problem of PCB components on sequential pick-and-place machines: mathematical models and heuristic solutions

William Ho[1,*] and Ping Ji[2]

[1]Operations and Information Management Group

Aston Business School, Aston University

Birmingham B4 7ET, United Kingdom

E-mail: w.ho@aston.ac.uk; Tel: +44 (0)121 2043342

[2]Department of Industrial and Systems Engineering,

The Hong Kong Polytechnic University,

Hung Hom, Kowloon, Hong Kong

E-mail: mfpji@polyu.edu.hk; Tel: +852 27666631

Abstract

This paper formulates several mathematical models for determining the optimal sequence of component placements and assignment of component types to feeders simultaneously or the integrated scheduling problem for a type of surface mount technology placement machines, called the sequential pick-and-place (PAP) machine. A PAP machine has multiple stationary feeders storing components, a stationary working table holding a printed circuit board (PCB), and a movable placement head to pick up components from feeders and place them to a board. The objective of integrated problem is to minimize the total distance traveled by the placement head. Two integer nonlinear programming models are formulated first. Then, each of them is equivalently converted into an integer linear type. The models for the integrated problem are verified by two commercial packages. In addition, a hybrid genetic algorithm previously developed by the authors is adopted to solve the models. The algorithm not only generates the optimal solutions quickly for small-sized problems, but also outperforms the genetic algorithms developed by other researchers in terms of total traveling distance.

Keywords: Printed circuit board manufacturing; Surface mount technology; Component sequencing; Feeder arrangement; Mathematical modeling; Genetic algorithm

* Corresponding author.

1. Introduction

The wide applicability of PCB has driven researchers and manufacturers to concentrate on the PCB assembly process planning in order to improve efficiency and remain competitive. Process planning consists of two closely related issues: setup management and process optimization [1]. Setup management involves four sub-problems:

- *Line assignment*: assigning PCBs to different assembly lines;
- *Machine grouping*: grouping placement machines;
- *PCB grouping*: grouping PCBs into families;
- *PCB sequencing*: sequencing the production of PCBs.

On the other hand, process optimization includes three sub-problems:

- *Component allocation*: allocating component types to different placement machines;
- *Feeder arrangement*: arranging component types to different feeders at each machine;
- *Component sequencing*: determining the component placement sequence.

The focus of this paper is confined to integrating the second and third sub-problems of process optimization for the sequential pick-and-place (PAP) machines. The performance of PAP machine is dependent on both component sequencing (i.e., which component is placed first, second, and so on) and feeder arrangement (i.e., which feeder stores which type of components). If the arrangement of components to feeders is not made carefully, even the pick and placement sequencing is solved for optimality, it can result in an extremely poor performance [2]. So, certainly, the component sequencing and feeder arrangement problems should be solved simultaneously.

2. Literature review

It is, however, observed that many researchers solved the component sequencing and feeder arrangement problems for the PAP machine separately. Ball and Magazine [3] modeled the component sequencing problem for the sequential PAP machine as the rural postman problem. The assumption was that the feeder arrangement was given. A heuristic approach was then used to solve the problem, which assured the solution to be optimal if the movement of assembly head was rectilinear. Ji *et al.* [4] formulated the component sequencing and feeder arrangement problems for the PAP machine as two individual linear assignment problems, each of which was solved with the reduced matrix method. Foulds and Hamacher [5] determined the sequence of component placements and feeder arrangement for the PAP machine sequentially. The feeder arrangement problem, which was formulated as a number of one-facility location models, was solved first. Then, the component sequencing problem, which was formulated as the traveling salesman problem (TSP), was solved after. Francis *et al.* [6] formulated the component sequencing and feeder arrangement problems for the PAP machine as the TSP with a special structure in order to minimize the total assembly time. A heuristic method called the 'clock sequence' was developed to solve the problem. Once the sequence of component placements was obtained, the feeder arrangement problem was also known with reference to the placement sequence. Kumar and Li [7] determined the sequence of component placements and feeder arrangement for the PAP machine. Although they formulated the problems as an integer quadratic programming model, they did not solve it to optimality because they found that the model was computationally intractable. The authors solved the problems separately instead. First, the component sequencing problem was referred as the TSP. The nearest neighbor, the nearest insertion, the furthest insertion, and the random generation were used to generate an initial sequence. Then, the 2-optimality, the 3-optimality, and the or-optimality heuristics were used to improve the initial sequence. Second, the feeder arrangement problem was referred as the minimum weight matching problem (MWMP). They used commercial package to obtain an optimal solution for the MWMP. Broad *et al.* [8] formulated the component sequencing and feeder arrangement problems for the PAP machine as an integer linear programming model. They adopted the solution procedure, which was developed by other researchers, to solve the feeder arrangement problem first and then the component sequencing problem. Magyar *et al.* [9] applied several search heuristics to solve the component sequencing and feeder arrangement problems separately for the PAP machine. They first determined the feeder arrangement, and then the order of component placements.

Only few researchers studied both component sequencing and feeder arrangement problems simultaneously (i.e., the integrated problem) for the PAP machine. Surprisingly, all of them applied genetic algorithms (GAs) to find near-optimal solution to the integrated problem [10-12]. This is mainly due to the success of GAs in solving a wide variety of complex optimization problems such as the TSP and quadratic assignment problem (QAP) [13-14], and the advantages of GAs such as simplicity, easy operation, and great flexibility.

Mathematical modeling is a powerful tool. Without applying it, the optimal solution to a particular problem cannot be obtained. Although heuristic methods like GAs are alternative tools for solving the problem, no one can guarantee that the solution generated is optimal or even no one knows how good the solution is before the optimal solution has been found. According to the literature review, only Kumar and Li [7] and Broad *et al.* [8] formulated mathematical models for the integrated problem. Kumar and Li [7] presented a nonlinear programming model to the problem. They, however, did not consider the starting point or home position of placement head. Besides, they did not verify and solve the model. Broad *et al.* [8] did not incorporate sub-tour elimination constraint in their model. The solution to their model may be infeasible. To overcome the inadequacies, this paper not only formulates mathematical models for the integrated problem but also verifies them by two commercial packages.

3. The sequential pick-and-place machine

This paper focuses on the sequential PAP machine. In this type of machines, the components are stored in multiple stationary feeders. The placement head travels to pick up a component from a feeder at a time, and then place it on the stationary board. The PAP machine is able to achieve high accuracy, and suitable to operate with large components such as integrated circuits (IC). The operation sequence of PAP machine is that the placement head starts from its original location (starting point or home position), moves to a feeder that carries components, picks up a component from the feeder, then moves to the desired placement location on the stationary board, and places it there. After that, the head moves back to the previous feeder if the next component is the same type with the previous one or moves to another feeder to pick up the next component if it is different from the previous one and place it on the board. The head repeats this operation procedure until all components are placed on the board and returns to its starting point, waits for the next board, as illustrated in Fig. 1 for 10 components in a board.

Consider a PCB to be assembled by a PAP machine. The PCB has n components with μ different types. Each component type can be stored in any feeder, and a feeder can only store a unique type of components, so μ feeders are required. The objective of integrated problem is to minimize the total traveling distance of placement head, which includes the distance from the starting point to a feeder at the beginning (i.e., d_{0l}), the distances from a feeder to a component's position on the PCB (i.e., d_{lj}), the distances from a component's position to a feeder (i.e., d_{il}), and the distance from the last component's position to the starting point (i.e., d_{i0}). The notation used in both individual and integrated mathematical models is summarized in Table 1.

4. Individual mathematical models

4.1 *A component sequencing model*

Suppose the assignment of component types to feeders (i.e., the feeder arrangement problem) is solved beforehand, the component sequencing model can then be formulated for finding the optimal total traveling distance of placement head. In order to achieve this goal, a decision variable is defined as:

$$x_{ij} = \begin{cases} 1 & \text{if component } i \text{ is placed immediately before component } j, \\ 0 & \text{otherwise.} \end{cases}$$

Actually, the component sequencing problem is somewhat similar to the TSP except the objective function. For the PAP machine, the objective is not to minimize the distance between component *i* and component *j* because the placement head is unable to place the next component on the PCB immediately without picking up a component from a feeder first. Therefore, the objective for the PAP machine should be to minimize the summation of different distances including:

- The distance between the position of component *i* on the PCB and feeder *l* (if *i* = 0, it is the distance between the starting point at the beginning and feeder *l*);
- The distance between feeder *l* and the position of the next component *j*;
- The distance between the position of the last component *i* and the starting point at the end.

An individual mathematical model for the component sequencing problem can be formulated as:

$$\text{Minimize } z = \sum_{i=0}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \sum_{l=1}^{\mu} \left(d_{il} + d_{lj}\right) x_{ij} + \sum_{i=1}^{n} d_{i0}\, x_{i0} \tag{1}$$

subject to

$$\sum_{i=0}^{n} x_{ij} = 1 \qquad \text{for } j = 0, 1, \ldots, n;\ i \neq j. \tag{2}$$

$$\sum_{j=0}^{n} x_{ij} = 1 \qquad \text{for } i = 0, 1, \ldots, n;\ i \neq j. \tag{3}$$

$$u_i - u_j + n x_{ij} \leq n - 1 \qquad \text{for } i, j = 1, 2, \ldots, n;\ i \neq j. \tag{4}$$

All x_{ij} = 0 or 1, All $u_i \geq 0$ and is a set of integers (M1)

In M1, the objective function (1) is to minimize the total traveling distance of placement head. If the moving speed of placement head is incorporated, then the objective can be to minimize the total placement time for assembling all components on a PCB. Constraint set (2) ensures that exactly one component must be placed immediately before component *j*. Constraint set (3) ensures that exactly one component must be placed immediately after component *i*. These two constraint sets, however, are not sufficient. Although the solution drawn satisfies both constraint sets (2) and (3), it may still be infeasible due to the occurrence of sub-tours. Constraint set (4) is, therefore, added in order to eliminate sub-tours. Since the starting point must be visited first, it is redundant to include *i* and/or $j = 0$ in constraint set (4).

4.2 *A feeder arrangement model*

If the component placement sequence or x_{ij} in M1 is known, we need to arrange a component type to a feeder, and this is the second problem to be studied in this paper, called the feeder arrangement problem. It is to assign the component types to feeders in such a way that the total distance traveled by the placement head is minimized. In order to achieve this goal, a decision variable is defined as:

$$y_{t_j l} = \begin{cases} 1 & \text{if component type } t \text{ of component } j \text{ is stored in feeder } l, \\ 0 & \text{otherwise.} \end{cases}$$

An individual mathematical model for the feeder arrangement problem can be formulated as:

$$\text{Minimize } z = \sum_{i=0}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} \sum_{t=1}^{\mu} \sum_{l=1}^{\mu} \left(d_{il} + d_{lj}\right) y_{t_j l} + d_{i0} \tag{5}$$

subject to

$$\sum_{t=1}^{\mu} y_{tl} = 1 \qquad \text{for } l = 1, 2, \ldots, \mu. \tag{6}$$

$$\sum_{l=1}^{\mu} y_{tl} = 1 \qquad \text{for } t = 1, 2, \ldots, \mu. \tag{7}$$

All $y_{tl} = 0$ or 1 (M2)

In M2, the objective function (5) is to calculate the total distance traveled by the placement head for assembling all components. Constraint set (6) ensures that exactly one component type is stored in one feeder. Constraint set (7) ensures that exactly one feeder is used to store one component type. The mathematical model for the feeder arrangement problem is somewhat similar to the QAP except the objective function.

5. Integrated mathematical models

5.1 *Formulation 1*

Before solving M1, it is essential to obtain the solution of feeder arrangement problem or M2 first. On the other hand, M2 cannot be solved until the solution of component sequencing problem or M1 is known. It is no doubt that the component sequencing and feeder arrangement problems are inter-related. Moreover, the objective function in M1 and M2 is to minimize the total distance traveled by the placement head. It can be noticed that the amount of distance traveled is dependent on the position of the next component to place together with which feeder stores the next component to pick up. In order to obtain the optimal solution, both problems should, therefore, be considered and solved simultaneously. Otherwise, it can result in poor performance [2].

A pure integer nonlinear programming model for the integrated problem can be formulated as:

$$\text{Minimize } z = \sum_{i=0}^{n} \sum_{\substack{j=1 \\ j\neq i}}^{n} \sum_{t=1}^{\mu} \sum_{l=1}^{\mu} \left(d_{il} + d_{lj}\right) x_{ij}\, y_{t_j l} + \sum_{i=1}^{n} d_{i0}\, x_{i0} \tag{8}$$

subject to

$$\sum_{i=0}^{n} x_{ij} = 1 \qquad \text{for } j = 0, 1, \ldots, n;\ i \neq j. \tag{9}$$

$$\sum_{j=0}^{n} x_{ij} = 1 \qquad \text{for } i = 0, 1, \ldots, n;\ i \neq j. \tag{10}$$

$$u_i - u_j + n x_{ij} \leq n - 1 \qquad \text{for } i, j = 1, 2, \ldots, n;\ i \neq j. \tag{11}$$

$$\sum_{t=1}^{\mu} y_{tl} = 1 \qquad \text{for } l = 1, 2, \ldots, \mu. \tag{12}$$

$$\sum_{l=1}^{\mu} y_{tl} = 1 \qquad \text{for } t = 1, 2, \ldots, \mu. \tag{13}$$

All x_{ij} and y_{tl} = 0 or 1, All $u_i \geq 0$ and is a set of integers (M3)

Since there is nonlinear term $x_{ij}\, y_{t_j l}$ in the objective function and the model contains both binary variables (i.e., x_{ij} and $y_{t_j l}$) and integer variables (i.e., u_i and u_j), M3 can be regarded as a pure integer nonlinear programming model. The objective function (8) calculates the total traveling distance of placement head, whereas the interpretation of constraint sets (9) to (13) was mentioned in M1 and M2.

Since M3 only contains nonlinear function in the form of products of binary variables, it can be reformulated as a linear programming model by implementing the following steps:

- Introduce a new binary variable w to replace the product term xy;
- Make use of the extra constraints: $w \le x$, $w \le y$, and $w \ge x + y - 1$ to reflect the logical condition: $w = 1$ if and only if $x = 1$ and $y = 1$.

The nonlinear term in the objective function is in the form of products of two binary variables. It can, therefore, be rewritten as a linear one by introducing an extra binary variable w_{ijl} as well as three extra constraint sets. The interpretation of decision variable w_{ijl} is:

$$w_{ijl} = \begin{cases} 1 & \text{if component } j \text{ is placed just after component } i \text{ and} \\ & \text{the type of component } j \text{ is stored in feeder } l, \\ 0 & \text{otherwise.} \end{cases}$$

M3 can be converted into a pure integer linear programming model as follows:

$$\text{Minimize } z = \sum_{i=0}^{n} \sum_{\substack{j=1 \\ j \ne i}}^{n} \sum_{l=1}^{\mu} \left(d_{il} + d_{lj}\right) w_{ijl} + \sum_{i=1}^{n} d_{i0}\, x_{i0} \qquad (14)$$

subject to

$$\sum_{i=0}^{n} x_{ij} = 1 \qquad \text{for } j = 0, 1, \ldots, n;\ i \ne j. \qquad (15)$$

$$\sum_{j=0}^{n} x_{ij} = 1 \qquad \text{for } i = 0, 1, \ldots, n;\ i \ne j. \qquad (16)$$

$$u_i - u_j + n x_{ij} \le n - 1 \qquad \text{for } i, j = 1, 2, \ldots, n;\ i \ne j. \qquad (17)$$

$$\sum_{t=1}^{\mu} y_{tl} = 1 \qquad \text{for } l = 1, 2, \ldots, \mu. \qquad (18)$$

$$\sum_{l=1}^{\mu} y_{tl} = 1 \qquad \text{for } t = 1, 2, \ldots, \mu. \qquad (19)$$

$$w_{ijl} \le x_{ij} \qquad \text{for } i = 0, 1, \ldots, n; \text{ for } j = 1, 2, \ldots, n;\ i \ne j; \text{ for } l = 1, 2, \ldots, \mu. \qquad (20)$$

$$w_{ijl} \le y_{t_j l} \qquad \text{for } i = 0, 1, \ldots, n; \text{ for } j = 1, 2, \ldots, n;\ i \ne j; \text{ for } l, t = 1, 2, \ldots, \mu. \qquad (21)$$

$$w_{ijl} \ge x_{ij} + y_{t_j l} - 1 \qquad \text{for } i = 0, 1, \ldots, n; \text{ for } j = 1, 2, \ldots, n;\ i \ne j; \text{ for } l, t = 1, 2, \ldots, \mu. \qquad (22)$$

All x_{ij}, y_{tl}, and w_{ijl} = 0 or 1, All $u_i \ge 0$ and is a set of integers (M4)

In M4, the objective function (14), and the constraint sets (20) to (22) are the linear expression of objective function (8) in M3. The interpretation of constraint sets (15) to (19) in M4 is the same as that of constraint sets (9) to (13) in M3.

5.2 *Formulation 2*

Although the constraint sets (11) and (17) are able to guarantee that the solution generated is feasible, they increase the complexity of models as there are $n(n-1)$ constraints in this sub-tour elimination constraint. In order to reduce the burden of models, it is essential to find a way to replace the bulky constraint. In this paper, M3 is re-modeled or the constraint set (11) in M3 is omitted using another decision variable x_{ip} instead of x_{ij}. The interpretation of x_{ip} is that:

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

The idea is to assign n components to n positions or placement orders, which means that there are totally n^2 decision variables in which only n variables are 1 while all others are 0. Since each component must be placed in exactly one position, no sub-tour will be appeared in this situation. Referring to Fig. 1, x_{21} and x_{32} are both 1 because component 2 and component 3 are placed first and second, respectively. A binary integer nonlinear programming model for the integrated problem can be formulated as follows:

$$\text{Minimize } z = \sum_{j=1}^{n}\sum_{t=1}^{\mu}\sum_{l=1}^{\mu}\left(d_{0l} + d_{lj}\right)x_{j1}\, y_{t_j l}$$

$$+ \sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n}\sum_{p=1}^{n-1}\sum_{t=1}^{\mu}\sum_{l=1}^{\mu}\left(d_{il} + d_{lj}\right)x_{ip}\, x_{j,p+1}\, y_{t_j l} + \sum_{i=1}^{n} d_{i0}\, x_{in} \qquad (23)$$

subject to

$$\sum_{i=1}^{n} x_{ip} = 1 \qquad \text{for } p = 1, 2, \ldots, n. \qquad (24)$$

$$\sum_{p=1}^{n} x_{ip} = 1 \qquad \text{for } i = 1, 2, \ldots, n. \qquad (25)$$

$$\sum_{t=1}^{\mu} y_{tl} = 1 \qquad \text{for } l = 1, 2, \ldots, \mu. \qquad (26)$$

$$\sum_{l=1}^{\mu} y_{tl} = 1 \qquad \text{for } t = 1, 2, \ldots, \mu. \qquad (27)$$

All x_{ip} and y_{tl} = 0 or 1 (M5)

In M5, the objective function (23) calculates the total distance traveled by the placement head. Constraint set (24) ensures that exactly one component is placed in one position. Constraint set (25) ensures that one position has exactly one component placed. Constraint set (26) ensures that exactly one component type is stored in one feeder. Constraint set (27) ensures that exactly one feeder is used to store one component type.

Similarly, M5 can be reformulated to a linear programming model. In the objective function (23) of M5, the first nonlinear term (i.e., $x_{j1}\ y_{t_j l}$) is in the form of products of two binary variables. It can, therefore, be rewritten as a linear term by introducing an extra binary variable w_{j1l} as well as three extra constraint sets. The interpretation of decision variable w_{j1l} is:

$$w_{j1l} = \begin{cases} 1 & \text{if component } j \text{ is placed first and} \\ & \text{the type of component } j \text{ is stored in feeder } l, \\ 0 & \text{otherwise.} \end{cases}$$

In the objective function (23) of M5, the second nonlinear term (i.e., $x_{ip}\ x_{j,p+1}\ y_{t_j l}$) is in the form of products of three binary variables. So, the steps for converting it into linear type need to be modified in this case. The major difference is that four instead of three extra constraints are introduced. Similarly, a decision variable $w_{ij(p+1)l}$ is introduced. The decision variable $w_{ij(p+1)l}$ is defined as:

$$w_{ij(p+1)l} = \begin{cases} 1 & \text{if component } j \text{ is placed just after component } i \text{ and} \\ & \text{the type of component } j \text{ placed in the } (p+1)\text{th position is stored in feeder } l, \\ 0 & \text{otherwise.} \end{cases}$$

M5 can be converted into a binary integer linear programming model as follows:

$$\text{Minimize } z = \sum_{j=1}^{n}\sum_{t=1}^{\mu}\sum_{l=1}^{\mu}\left(d_{0l} + d_{lj}\right) w_{j1l}$$

$$+ \sum_{i=1}^{n}\sum_{\substack{j=1 \\ j\neq i}}^{n}\sum_{p=1}^{n-1}\sum_{t=1}^{\mu}\sum_{l=1}^{\mu}\left(d_{il} + d_{lj}\right) w_{ij(p+1)l} + \sum_{i=1}^{n} d_{i0}\ x_{in} \tag{28}$$

subject to

$$\sum_{i=1}^{n} x_{ip} = 1 \qquad \text{for } p = 1, 2, \ldots, n. \tag{29}$$

$$\sum_{p=1}^{n} x_{ip} = 1 \qquad \text{for } i = 1, 2, \ldots, n. \tag{30}$$

$$\sum_{t=1}^{\mu} y_{tl} = 1 \qquad \text{for } l = 1, 2, \ldots, \mu. \tag{31}$$

$$\sum_{l=1}^{\mu} y_{tl} = 1 \qquad \text{for } t = 1, 2, \ldots, \mu. \tag{32}$$

$$w_{j1l} \le x_{j1} \qquad \text{for } j = 1, 2, \ldots, n; \; \text{for } l = 1, 2, \ldots, \mu. \tag{33}$$

$$w_{j1l} \le y_{t_j l} \qquad \text{for } j = 1, 2, \ldots, n; \; \text{for } l, t = 1, 2, \ldots, \mu. \tag{34}$$

$$w_{j1l} \ge x_{j1} + y_{t_j l} - 1 \qquad \text{for } j = 1, 2, \ldots, n; \; \text{for } l, t = 1, 2, \ldots, \mu. \tag{35}$$

$$w_{ij(p+1)l} \le x_{ip} \qquad \text{for } i, j = 1, 2, \ldots, n;\; i \ne j; \; \text{for } p = 1, 2, \ldots, n-1; \; \text{for } l = 1, 2, \ldots, \mu. \tag{36}$$

$$w_{ij(p+1)l} \le x_{j,p+1} \qquad \text{for } i, j = 1, 2, \ldots, n;\; i \ne j; \; \text{for } p = 1, 2, \ldots, n-1; \; \text{for } l = 1, 2, \ldots, \mu. \tag{37}$$

$$w_{ij(p+1)l} \le y_{t_j l} \qquad \text{for } i, j = 1, 2, \ldots, n;\; i \ne j; \; \text{for } p = 1, 2, \ldots, n-1; \; \text{for } l, t = 1, 2, \ldots, \mu. \tag{38}$$

$$w_{ij(p+1)l} \ge x_{ip} + x_{j,p+1} + y_{t_j l} - 2 \qquad \text{for } i, j = 1, 2, \ldots, n;\; i \ne j; \; \text{for } p = 1, 2, \ldots, n-1; \; \text{for } l, t = 1, 2, \ldots, \mu. \tag{39}$$

All x_{ip}, y_{tl}, w_{j1l}, and $w_{ij(p+1)l}$ = 0 or 1 (M6)

In M6, the constraint sets (33) to (35), and the first term in the objective function (28) are the linear expression of the first term in the objective function (23) of M5. Besides, the constraint sets (36) to (39), and the second term in the objective function (28) are the linear expression of the second term in the objective function (23) of M5. The interpretation of constraint sets (29) to (32) in M6 is the same as that of constraint sets (24) to (27) in M5.

6. Computational analysis

In this section, the complexity of four integrated mathematical models (i.e., M3 to M6) is discussed and compared first. Then, the models with less variables and constraints are solved to optimality.

In order to examine the complexity of models, it is essential to find out the numbers of variables and constraints in each model. According to M3, it can be seen that the model is very sophisticated and hard to solve. Not only the objective function is nonlinear, but also its possible enumeration is huge. M3 has ($n^2 + n + \mu^2$) binary variables, n integer variables, and ($n^2 + n + 2\mu + 2$) constraints. In the objective function (8), the possible terms are $n\mu + n\mu(n - 1) + n$ or $n^2\mu + n$. For M4, although the model becomes linear, both numbers of variables and constraints increase greatly at the same time. Both numbers in M4 are much greater than that in M3. For the number of variables, $n^2\mu$ of w_{ijl} are introduced in M4 besides ($n^2 + n + \mu^2$) binary variables and n integer variables. For the number of constraints, besides ($n^2 + n + 2\mu + 2$) constraints, M4 has $3n^2\mu$ constraints more for the constraint sets (20) to (22).

After adopting another decision variable in M5, the bulky sub-tour elimination constraint is omitted. The complexity of M5 is lower than that of M3 as both numbers of variables and constraints reduce significantly. M5 has only ($n^2 + \mu^2$) binary variables, and ($2n + 2\mu$) constraints. Since there are two nonlinear terms in the objective function (23) in M5, two additional decision variables and seven extra constraints are necessary to be incorporated in the equivalent linear programming model or M6. As a result, M6 becomes enormous and very complex. Both numbers of variables and constraints are even much greater than that in M4. For the number of binary variables, $n\mu$ of w_{j1l} and $n(n - 1)^2\mu$ of $w_{ij(p+1)l}$ are introduced in M6 besides ($n^2 + \mu^2$) binary variables. For the number of constraints, besides ($2n + 2\mu$) constraints, M6 has $3n\mu + 4n(n - 1)^2\mu$ constraints more in which there are $3n\mu$ constraints for the constraint sets (33) to (35) and $4n(n - 1)^2\mu$ constraints for the constraint sets (36) to (39). The numbers of variables and constraints of four models are listed in Table 2.

For a realistically sized problem of 100 components and 10 component types, M3 has 10,300 variables and 10,122 constraints. Comparatively, M5 is a better nonlinear programming formulation because it consists of 10,100 variables together with 220 constraints merely. For the linear programming formulation, M4 is much more desirable than M6. M4 has 110,300 variables and 310,122 constraints. Both numbers of variables and constraints in M6, however, are much greater. It has 9,812,100 variables as well as 39,207,220 constraints! So, the model may not be solved to optimality in a reasonable amount of time.

Due to the fact that M4 and M5 are the better linear and nonlinear formulations in terms of complexity, respectively, these two models are solved to global optimality. To solve the models, two commercial packages are used. First, CPLEX is a well-known powerful integer linear programming solver. It is, therefore, applied to solve M4. Second, BARON is a computational system for solving non-convex optimization problems including binary integer nonlinear programming models to global optimality. So, it is adopted to solve M5. By these two commercial packages, the models are tested by several small examples, and both have the same solutions to the same examples.

According to Table 3, it is found that the pure integer linear programming model (i.e., M4) is more desirable than the binary integer nonlinear programming model (i.e., M5) in terms of amount of computational time spent. For instance, it spends 10 and a half hours by CPLEX to solve M4 with 8 components and 8 types to optimality. But, it takes more than 15 days by BARON to solve M5 with the same problem size to optimality. Since the optimal solutions of M4 can be obtained more quickly, an integrated problem with 10 components and 6 types is formulated as the pure integer linear programming model or M4 for getting the optimal solution. The data of problem is listed in Table 4. The optimal assembly sequence of placement head is: starting point $\rightarrow f_3 \rightarrow c_2 \rightarrow f_3 \rightarrow c_3 \rightarrow f_2 \rightarrow c_4 \rightarrow f_1 \rightarrow c_5 \rightarrow f_1 \rightarrow c_{10} \rightarrow f_2 \rightarrow c_9 \rightarrow f_6 \rightarrow c_6 \rightarrow f_5 \rightarrow c_7 \rightarrow f_5 \rightarrow c_8 \rightarrow f_4 \rightarrow c_1 \rightarrow$ starting point, whereas the total distance traveled by the placement head is 566.02 mm.

Although the global optimum of both models can be obtained, they are not efficient approaches since the computational time grows exponentially with the problem size. In addition, the number of components on a PCB in the real-world situations is quite large, normally several hundreds. It is unacceptable to spend several days or even hours to solve the integrated problem. To solve the problem efficiently, a heuristic method should be adopted.

7. A hybrid genetic algorithm

GA, developed by John Holland in the 1960s, is a stochastic optimization technique. Similar to simulated annealing (SA) and tabu search (TS), GA can avoid getting trapped in a local optimum by the aid of mutation operation. Actually, the basic idea of GA is to maintain a population of candidate solutions that evolves under a selective pressure. Hence, it can be viewed as a class of local search based on a solution-generation mechanism operating on attributes of a set of solutions rather than attributes of a single solution by the move-generation mechanism of the local search methods, like SA and TS [15].

Since the component sequencing and feeder arrangement problems are considered simultaneously, a simple GA may not perform well in this situation. It was proved that the individual problems are already very hard to solve [16]. The GA adopted here is, therefore, hybridized with several heuristics in order to improve the solution further. It is, however, found that none of the previous researchers used the hybrid GA or HGA to solve the integrated problem for the PAP machine.

The flowchart of HGA for the integrated problem is shown in Fig. 2. Since both problems are considered simultaneously, each chromosome or solution includes two path representations or links (Fig. 3). The first link denotes the component sequencing, whereas the second link represents the feeder arrangement. After the parameters (i.e., the population size, iteration number, crossover rate, and mutation rate) have been set up, the HGA generates an initial population in which the first links are generated from the nearest neighbor heuristic while the second links are generated randomly. During this initialization step, each chromosome is improved as follows: the iterated swap procedure (ISP) (Fig. 4) is performed on the first link while the 2-opt local search heuristic is applied to the second link. Actually, the principle of ISP is very similar to that of the 2-opt local search heuristic, except that some instead of all two swaps are examined to generate offspring. It can definitely reduce the computational time because the number of components is quite large, normally several hundreds. Each chromosome is then measured by an evaluation function, which was described thoroughly in Section 5. The roulette wheel selection operation [13] is performed to select some chromosomes for the genetic operations including the modified order crossover (Fig. 5), the heuristic mutation (Fig. 6), and the inversion mutation (Fig. 7). After an offspring is produced, the first link is improved by the ISP while the second link is improved by the 2-opt local search heuristic. The fitness of offspring will be measured and may become a member of population if it possesses a relatively good quality. These steps form an iteration, and then the roulette wheel selection is performed again to start the next iteration. The HGA will not stop unless the predetermined number of iterations is conducted.

8. Performance analysis

In this section, the performance of HGA is evaluated by comparing it to the optimal solutions of several problems mentioned in Section 6 first. Then, the comparison between the HGA and the GAs developed by Leu *et al.* [10] and Ong and Khoo [11] is carried out. It is unable to compare with the GA proposed by Loh *et al.* [12] because there is no data provided.

8.1 *Comparison to optimal solutions*

The same problems as mentioned in Section 6 are solved by the HGA. It is found that the HGA achieves the optimal solutions to all problems quickly. Furthermore, the longest computational time spent is only 9 seconds for the 8-component problem. Compared with the time spent on finding the optimal solution for the 8-component problem, the HGA is much more efficient. It saves more than 10 hours when compared with CPLEX, and saves about 15 days when compared with BARON.

8.2 *Comparison to other approaches*

The performance of HGA is evaluated using the PCB example (refer to Leu *et al.* [10]) in which there are 200 components and 10 component types. The HGA parameters are set as: population size = 25, iteration number = 3000, crossover rate = 0.4, and mutation rate = 0.2. According to Table 5, it is found that the performance of HGA is superior to that of GAs [10- 11] in three aspects. Firstly, the HGA can obtain a better solution with a smaller population size, 25 only, while the other two used 100. Secondly, the HGA can obtain a better solution not only with a smaller population size, but also with a fewer iterations, 3,000 vs. 6,150. Finally, and most importantly, the HGA obtained a better solution than any previous methods, 5,660.5 cm vs. 5,673.7 cm or 6,129 cm. If the traveling speed of placement head is assumed to be 60 mm/s, the improvement is 2.2 seconds when compared with Ong and Khoo [11]. Since the component placement is the bottleneck of PCB assembly line [17-18], a minor reduction in the cycle time will save a significant production time. For example, to produce 100,000 boards, a reduction of 2.2 seconds in the cycle time will save 3,667 minutes or 61 working hours. The productivity of a PCB manufacturing company can, therefore, be enhanced if our HGA is adopted.

In the above experiment, it is assumed that the number of feeders provided is exactly the same as the number of component types (i.e., 10). Each component type can only be stored in exactly one feeder. Herein, the 200-component problem is solved again using the HGA in which three additional feeders are available or there are totally 13 feeders. In this case, three types of components can be assigned to two feeders. In the 200-component problem, component types 2, 6, and 8 are the most frequently used, these three types of components can, therefore, be stored in two feeders. According to Table 5, it is noticed that the solution is much better (4855.5 cm) if there are additional feeders. Since the three most frequently used component types are assigned to more than one feeder, they can be retrieved from a closer feeder. So, the total distance traveled by the placement head can be reduced.

9. Conclusions

Mathematical modeling is a powerful tool in today's life. This paper presented several nonlinear and linear programming models for an integrated scheduling problem in a sequential pick-and-place machine. After the mathematical models for the individual component sequencing and feeder arrangement problems had been formulated, it was found that the problems are inter-related. One cannot be solved unless the solution of the other is obtained beforehand. Due to their close relationship, we formulated two integer nonlinear programming models and two integer linear programming models for solving the integrated problem for the sequential pick-and-place machine. In terms of amount of computational time spent, the model of linear type is more desirable.

Although the optimal solution can be found using the commercial packages, it was noticed that the computational time grows exponentially with the problem size. In order to solve the problem efficiently, a HGA was adopted. The algorithm is so called because the nearest neighbor heuristic, the iterated swap procedure, and the 2-opt local search heuristic are incorporated to improve the solution. It was shown that the performance of HGAs was superior to that of GAs proposed by previous researchers in terms of total traveling distance. Furthermore, the solution was even much better when component types could be stored in more than one feeder.

Acknowledgements

The work described in this paper was partially supported by a grant from the Research Grants Council of Hong Kong Special Administrative Region, China (Project No. PolyU 5259/04E).

References

[1] Ellis KP, Vittes FJ, Kobza JE. Optimizing the performance of a surface mount placement machine. IEEE Transactions on Electronics Packaging Manufacturing 2001; 24: 160-170.

[2] Altinkemer K, Kazaz B, Köksalan M, Moskowitz H. Optimization of printed circuit board manufacturing: integrated modeling and algorithms. European Journal of Operational Research 2000; 124: 409-421.

[3] Ball MO, Magazine MJ. Sequencing of insertions in printed circuit board assembly. Operations Research 1988; 36: 192-201.

[4] Ji Z, Leu MC, Wong H. Application of linear assignment model for planning of robotic printed circuit board assembly. Journal of Electronic Packaging 1992; 114: 455-460.

[5] Foulds LR, Hamacher HW. Optimal bin location and sequencing in printed circuit board assembly. European Journal of Operational Research 1993; 66: 279-290.

[6] Francis RL, Hamacher HW, Lee CY, Yeralan S. Finding placement sequences and bin locations for cartesian robots. IIE Transactions 1994; 26: 47-59.

[7] Kumar R, Li H. Integer programming approach to printed circuit board assembly time optimization. IEEE Transactions on Components, Packaging, and Manufacturing Technology – Part B 1995; 18: 720-727.

[8] Broad K, Mason A, Rönnqvist M, Frater M. Optimal robotic component placement. Journal of the Operational Research Society 1996; 47: 1343-1354.

[9] Magyar G, Johnsson M, Nevalainen O. On solving single machine optimization problems in electronics assembly. Journal of Electronics Manufacturing 1999; 9: 249-267.

[10] Leu MC, Wong H, Ji Z. Planning of component placement/insertion sequence and feeder setup in PCB assembly using genetic algorithm. Journal of Electronic Packaging 1993; 115: 424-432.

[11] Ong NS, Khoo LP. Genetic algorithm approach in PCB assembly. Integrated Manufacturing Systems 1999; 10: 256-265.

[12] Loh TS, Bukkapatnam STS, Medeiros D, Kwon H. A genetic algorithm for sequential part assignment for PCB assembly. Computers & Industrial Engineering 2001; 40: 293-307.

[13] Goldberg DE. Genetic algorithms in search, optimization and machine learning. New York: Addison-Wesley; 1989.

[14] Gen M, Cheng R. Genetic algorithms and engineering design. New York: Wiley; 1997.

[15] Osman IH, Kelly JP. Meta-heuristics: theory & applications. Boston: Kluwer Academic Publishers; 1996.

[16] Crama Y, Flippo OE, Klundert JVD, Spieksma FCR. The assembly of printed circuit boards: a case with multiple machines and multiple board types. European Journal of Operational Research 1997; 98: 457-472.

[17] Ong NS, Tan WC. Sequence placement planning for high-speed PCB assembly machine. Integrated Manufacturing Systems 2002; 13: 35-46.

[18] Wilhelm WE, Tarmy PK. Circuit card assembly on tandem turret-type placement machines. IIE Transactions 2003; 35: 627-645.

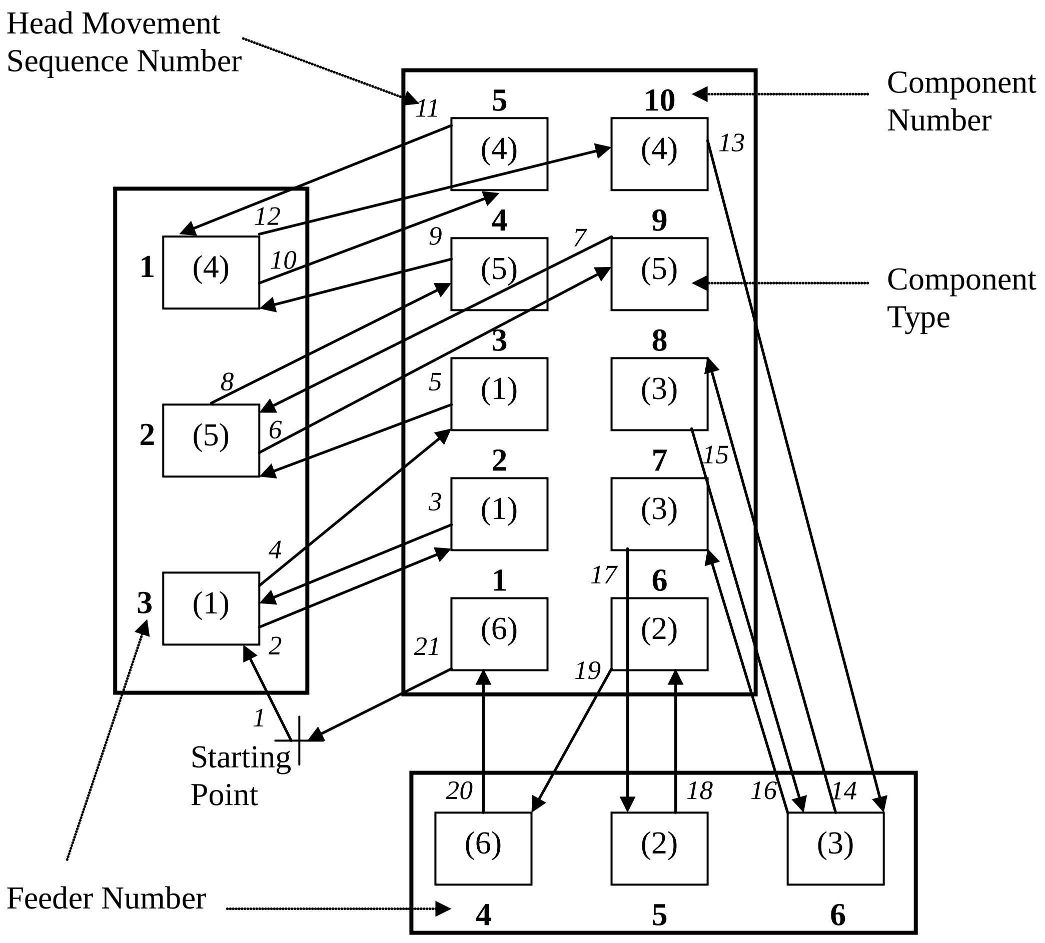

Fig. 1. The assembly sequence of placement head.

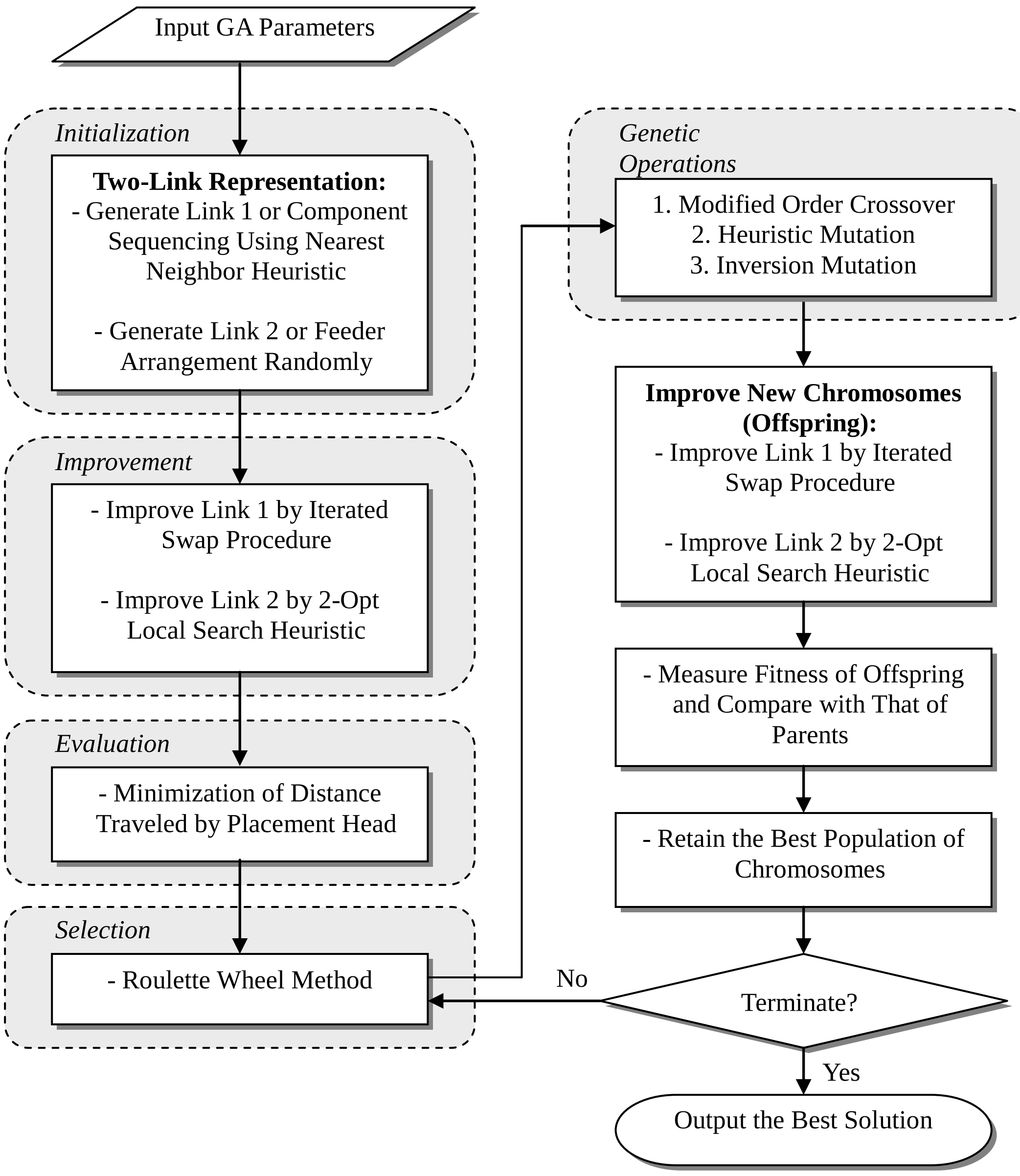
Input GA Parameters
Initialization
Two-Link Representation:
- Generate Link 1 or Component Sequencing Using Nearest Neighbor Heuristic
- Generate Link 2 or Feeder Arrangement Randomly
Improvement
- Improve Link 1 by Iterated Swap Procedure
- Improve Link 2 by 2-Opt Local Search Heuristic
Evaluation
- Minimization of Distance Traveled by Placement Head
Selection
- Roulette Wheel Method
Genetic Operations
1. Modified Order Crossover
2. Heuristic Mutation
3. Inversion Mutation
Improve New Chromosomes (Offspring):
- Improve Link 1 by Iterated Swap Procedure
- Improve Link 2 by 2-Opt Local Search Heuristic
- Measure Fitness of Offspring and Compare with That of Parents
- Retain the Best Population of Chromosomes
Terminate?
No
Yes
Output the Best Solution

Fig. 2. The flowchart of hybrid genetic algorithm.

Assembly Sequence	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Component Number	2	3	9	4	5	10	8	7	6	1

Link 1

Component Type	4	5	1	6	2	3
Feeder	**1**	**2**	**3**	**4**	**5**	**6**

Link 2

Fig. 3. The two-link representation for a chromosome.

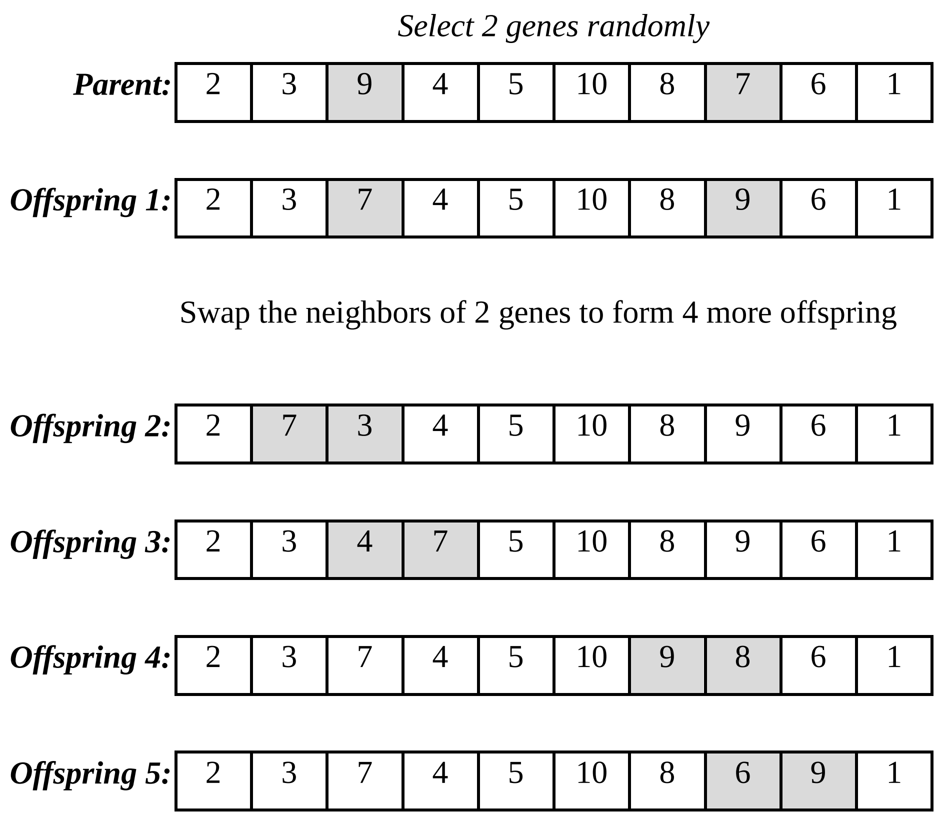
Select 2 genes randomly
Parent: 2 3 9 4 5 10 8 7 6 1
Offspring 1: 2 3 7 4 5 10 8 9 6 1
Swap the neighbors of 2 genes to form 4 more offspring
Offspring 2: 2 7 3 4 5 10 8 9 6 1
Offspring 3: 2 3 4 7 5 10 8 9 6 1
Offspring 4: 2 3 7 4 5 10 9 8 6 1
Offspring 5: 2 3 7 4 5 10 8 6 9 1

Fig. 4. The iterated swap procedure.

Selected sub-string

Parent 1:	2	3	9	**4**	**5**	**10**	**8**	7	6	1

Parent 2:	1	2	3	4	5	6	7	8	9	10

Proto-child:				**4**	**5**	**10**	**8**			

Find the gene right prior to the first gene of sub-string from the second parent, and place it in front of the sub-string in the proto-child.

Proto-child:			**3**	4	5	10	8			

Find the gene right behind the last gene of sub-string from the second parent, and place it just after the sub-string in the proto-child.

Proto-child:			3	4	5	10	8	**9**		

The remaining genes, that is, the genes not in the proto-child yet, form a sequence. Place the genes into the unfilled positions of proto-child from the left to the right according to the sequence in the second parent.

Offspring:	**1**	**2**	3	4	5	10	8	9	**6**	**7**

Repeat the steps above to produce the second offspring by exchanging the two parents

Fig. 5. The modified order crossover operator.

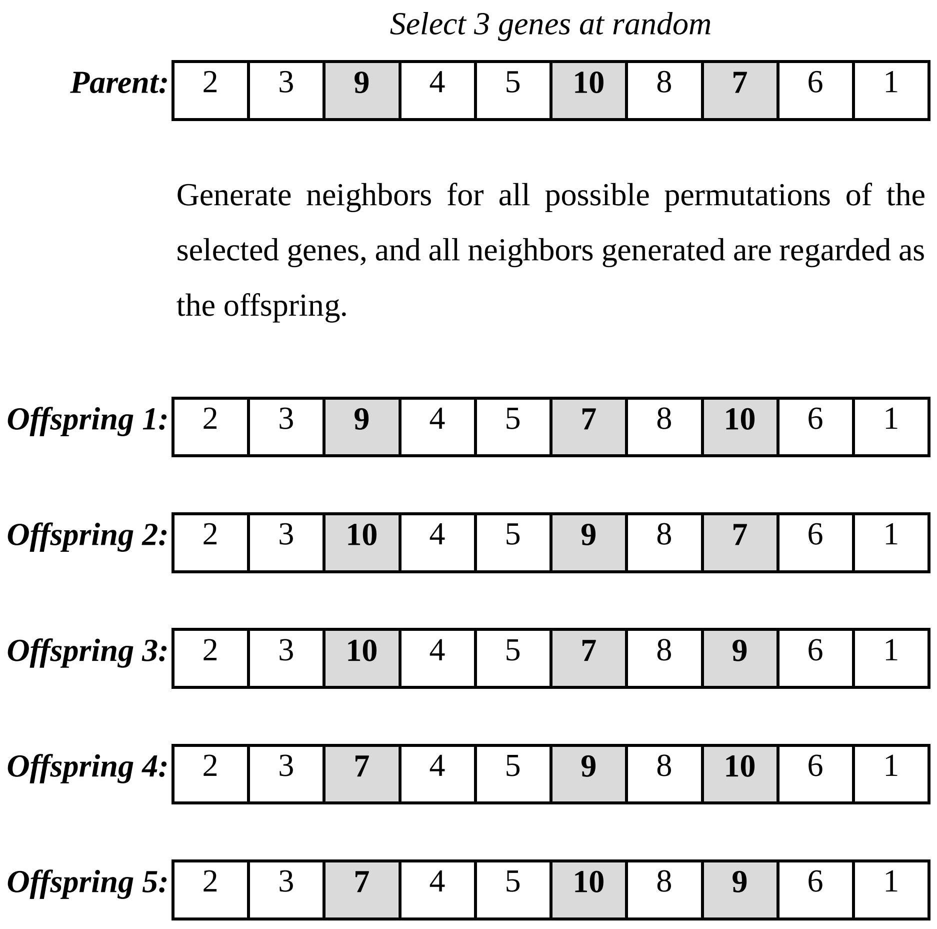
Select 3 genes at random
Parent: 2 3 **9** 4 5 **10** 8 **7** 6 1
Generate neighbors for all possible permutations of the selected genes, and all neighbors generated are regarded as the offspring.
Offspring 1: 2 3 **9** 4 5 **7** 8 **10** 6 1
Offspring 2: 2 3 **10** 4 5 **9** 8 **7** 6 1
Offspring 3: 2 3 **10** 4 5 **7** 8 **9** 6 1
Offspring 4: 2 3 **7** 4 5 **9** 8 **10** 6 1
Offspring 5: 2 3 **7** 4 5 **10** 8 **9** 6 1

Fig. 6. The heuristic mutation operator.

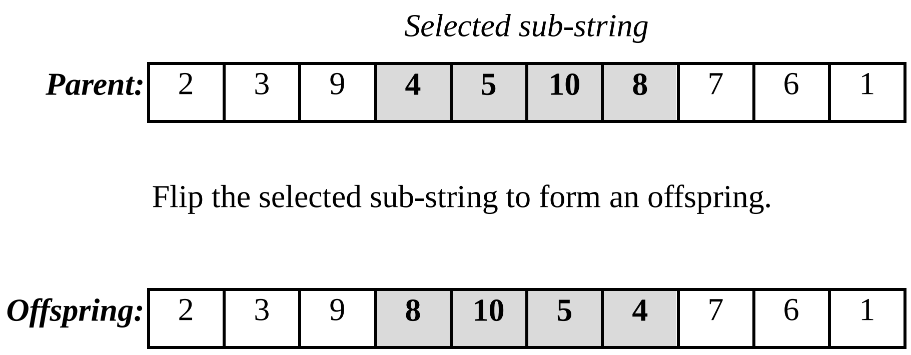

Fig. 7. The inversion mutation operator.

Table 1

Notation

Indices:

i, j: components ($i, j = 0, 1, \ldots, n$).

t: component types ($t = 1, 2, \ldots, \mu$).

l: feeders ($l = 1, 2, \ldots, \mu$).

p: placement order or placement position ($p = 1, 2, \ldots, n$).

Distances:

d_{0l}: distance traveled from starting point to feeder l.

d_{lj}: distance traveled from feeder l to the position of component j on the PCB.

d_{il}: distance traveled from the position of component i to feeder l.

d_{i0}: distance traveled from the position of component i to starting point.

Sub-tour elimination constraint:

u_i: placement order of component i.

Decision variables:

x_{ij} = 1 if component i is placed immediately before component j; 0 otherwise.

x_{ip} = 1 if component i is placed in the pth position; 0 otherwise.

$y_{t_j l}$ = 1 if component j with component type t is stored in feeder l; 0 otherwise.

Table 2

Numbers of variables and constraints in M3 to M6

	Number of variables	**Number of constraints**
M3	$n^2 + 2n + \mu^2$	$n^2 + n + 2\mu + 2$
M4	$(n^2 + 2n + \mu^2) + n^2\mu$	$(n^2 + n + 2\mu + 2) + 3n^2\mu$
M5	$n^2 + \mu^2$	$2n + 2\mu$
M6	$(n^2 + \mu^2) + n\mu + n(n - 1)^2\mu$	$(2n + 2\mu) + 3n\mu + 4n(n - 1)^2\mu$

Table 3

Computational time spent for solving M4 and M5

Numbers of components and types	**Optimal solution CPU time (hh:mm:ss) by CPLEX for M4**	**Optimal solution CPU time (hh:mm:ss) by BARON for M5**
4×4	0.16 seconds	0.31 seconds
5×5	00:00:01	00:00:04
6×6	00:00:41	00:01:52
7×7	00:02:26	01:21:39
8×8	10:30:51	379:02:50

Table 4

The data of integrated problem with 10 components and 6 types

Components	**Types**	**Coordinates (mm)**		**Feeders**	**Coordinates (mm)**	
		x	***y***		***x***	***y***
1	6	30	20	1	10	50
2	1	30	30	2	10	35
3	1	30	40	3	10	20
4	5	30	50	4	30	10
5	4	30	60	5	50	10
6	2	50	20	6	70	10
7	3	50	30			
8	3	50	40			
9	5	50	50			
10	4	50	60			

Table 5

A comparison of experimental results

	Leu *et al.* [10]	**Ong and Khoo [11]**	**HGA**	**HGA**
Number of extra feeders	N/A	N/A	N/A	3
Population size	100	100	25	25
Iteration number	6150	6150	3000	1000
Final best solution (cm)	about 6129	5673.7	5660.5	4855.5