Integrated component scheduling models for chip shooter machines

William Ho^{a,*}, Ping Ji^b

^aOperations 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

^bDepartment 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 focuses on minimizing printed circuit board (PCB) assembly time for a chip shooter machine, which has a movable feeder carrier holding components, a movable X-Y table carrying a PCB, and a rotary turret with multiple assembly heads. The assembly time of the machine depends on two inter-related optimization problems: the component sequencing problem and the feeder arrangement problem. Nevertheless, they were often regarded as two individual problems and solved separately. This paper proposes two complete mathematical models for the integrated problem of the machine. The models are verified by two commercial packages. Finally, a hybrid genetic algorithm previously developed by the authors is presented to solve the model. 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 assembly time.

Keywords: PCB assembly; Optimization; Component sequencing; Feeder arrangement; Hybrid genetic algorithm

* Corresponding author.

1. Introduction

The wide applicability of printed circuit boards (PCBs) has driven researchers or manufacturers to concentrate on the PCB assembly process planning in order to improve the efficiency and remain competitive. Besides, customers' contemporary requirements such as smaller in product size and greater in function and reliability force the surface mount technology (SMT) to replace the plated-through-hole (PTH) technology. Among several assembly operations in an SMT assembly line, the component placement is generally the most time-consuming. In addition, it is frequently a bottleneck of the line, and determines the line's cycle time (Wilhelm and Tarmy, 2003). Evidently, the throughput rate of a PCB manufacturing company or the company's competitiveness can be increased if the component placement process is optimized. In order to accomplish this goal, two inter-related optimization problems are considered for the chip shooter machine, which is one of the most commonly used SMT placement machines.

The component sequencing problem (i.e., which component is placed first, second, and so on) and the feeder arrangement problem (i.e., which component type is stored in which feeder) are the two inter-related optimization problems. For the PTH technology, the component sequencing problem of an auto-insertion machine can be simply formulated as the traveling salesman problem (TSP), and it is not necessary to consider the feeder arrangement problem (Chan and Mercier, 1989). However, for the SMT, the assembly time of a placement machine is also dependent on a feeder to hold which types of components besides the pick and placement sequence. If the arrangement of component types to feeders is not made carefully, even if the pick and placement sequencing is optimally solved, it can result in an extremely poor performance (Altinkemer et al., 2000). Since the chip shooter machine being studied in this paper is an SMT placement machine, the component sequencing and the feeder arrangement problems should be studied and solved simultaneously.

2. Literature review

For the chip shooter machine, many researchers solved the component sequencing and the feeder arrangement problems separately, and some efforts to solve the two problems together were also made.

2.1. The component sequencing problem

Souza and Wu (1995) studied the component sequencing problem only for the chip shooter machine. A knowledge-based component placement system incorporated with the TSP

algorithms was developed. Moyer and Gupta (1997) also formulated the component sequencing problem for the machine as the TSP, and developed the board sequencing heuristic to tackle the problem.

2.2. The feeder arrangement problem

Moyer and Gupta (1996a) formulated the feeder arrangement problem for the chip shooter machine as the quadratic assignment problem (QAP) based on the assumption that the sequence of component placements was predetermined. Two heuristic methods were proposed to solve the problem. Similarly, Dikos et al. (1997) formulated the feeder arrangement problem for the machine as the QAP, and made an assumption that an optimal component placement sequence was first specified. They employed genetic algorithms (GAs) to find a near optimal feeder arrangement. Klomp et al. (2000) treated the problem of determining an optimal feeder arrangement for a line of chip shooter machines as finding the shortest Hamiltonian path. Heuristic approaches were employed to deal with the problem.

2.3. The integrated component scheduling problem

Leu et al. (1993) and Ong and Tan (2003) applied GAs to solve the integrated problem, which is an integration of both component sequencing and feeder arrangement problems, simultaneously for the chip shooter machine. Bard et al. (1994) used an iterative approach to determine the component placement sequence, the feeder arrangement, and the retrieval plan for the machine separately. Moyer and Gupta (1996b) developed a heuristic algorithm for determining the component sequencing and the feeder arrangement problems individually. Sohn and Park (1996) formulated the component sequencing and the feeder arrangement problems as an integer nonlinear programming model for the machine while using the one-head case as an approximation to the multi-head case. A heuristic approach was developed to solve the problems sequentially. Yeo et al. (1996) developed a rule-based frame system to generate the feeder arrangement first for the machine, followed by the component placement sequence. Crama et al. (1997) also solved the feeder arrangement problem for the machine heuristically first, and then solved the component sequencing and the component retrieval problems using heuristic methods. Ellis et al. (2001) used heuristics to generate an initial placement sequence for the machine first, and then an initial feeder arrangement. The 2-opt local search heuristic was used to improve the initial solutions. In Wilhelm and Tarmy's approach (2003), each component type was assigned to a feeder of the machine first. After that, the sequence of component placements was determined by solving an asymmetric TSP.

One conclusion drawn from the above literatures is that a complete mathematical model for the integrated problem of the chip shooter machine has not been constructed yet. Although many heuristic approaches were developed, the effectiveness of these approaches is unknown because the optimal solution has not been found. Therefore, in Section 3 of this paper, two mathematical models are formulated for the integrated problem. Section 4 describes the hybrid GA for the problem briefly. Section 5 compares the result from the hybrid GA with the optimal solution, and with the genetic algorithms proposed by other researchers. Section 6 concludes the paper.

3. Mathematical models

A chip shooter machine, as illustrated in Fig. 1, has a movable feeder carrier holding components, a rotary turret with multiple assembly heads, and a movable X-Y table which locates a PCB. Each assembly head has several nozzles of different sizes. The operation sequence of the machine is as follows. As the first board of a batch enters the machine, the first nozzle of the assembly head picks up a component from a feeder. The turret then indexes one step and the next nozzle picks up the second component. The turret indexes again to pick up the next component, and so on. At the same time, the PCB is moved to the placement location waiting for the first component to be placed on the board. When the sixth component is being picked up, if the turret has 10 assembly heads, the first component is being placed on the board. These operations continue such as the turret indexes one step, the feeder carrier moves to the location containing the next pick-up component, and the X-Y table moves to the next placement location. In the assembly of the last five components, there is no need to pick up components for the board being assembled. However, the nozzles of the turret can pick up the first five components for the next board to be assembled if necessary. For the first few components assembled in a batch of PCBs, there are only pick-up movements and no placement movement. For the last few components of the same batch, there are only placement movements and no pick-up movement. These boundary effects can be neglected if the PCB's quantity in a batch is very large (Leu et al., 1993).

If there are g components between the pick-up and the placement locations, then there are $(2g + 2)$ assembly heads in the rotary turret. Each PCB has n components with μ different types. Besides, each of the component types must be stored in a feeder, but a feeder can only hold a unique type of components, so μ feeders are needed to hold μ types of components.

Three mechanisms of the machine move at different speeds, so the traveling time of the

X-Y table (i.e., $C1_{ij}$), the traveling time of the feeder carrier (i.e., $C2_{rs}$), and the indexing time of the turret (i.e., $C3$) are different from each other. These three times can be calculated as:

$C1_{ij}$ = time used by the X-Y table for traveling from component i (location) to component j (location).

$$= \max \left(\frac{|X_j - X_i|}{V_x}, \frac{|Y_j - Y_i|}{V_y} \right).$$

where X_i and X_j are the x -coordinates of the components i and j , respectively;

Y_i and Y_j are the y -coordinates of the components i and j , respectively;

V_x and V_y are speeds of the X-Y table in the x and y directions, respectively.

$C2_{rs}$ = time used by the feeder carrier for traveling from feeder r to feeder s .

$$= \frac{|X_s - X_r|}{V_f}.$$

where X_r and X_s are the x -coordinates of the feeders r and s , respectively;

V_f is the speed of the feeder carrier.

$C3$ = time used by the turret for rotating one step.

The longest one among the three times in one step is called the dominating time. So, the objective is to minimize the total assembly time, which is the summation of all dominating times of components such that the highest productivity of the machine can be achieved.

In the following sub-sections, two individual mathematical models for the component sequencing and the feeder arrangement problems are formulated first. Then, two complete mathematical models are constructed for the integrated problem. The notation used in the models is summarized in Table 1.

3.1. A component sequencing model

Assume the feeder arrangement problem is solved beforehand, the component sequencing problem is frequently formulated as the TSP for finding the placement order of components on a PCB such that the total traveling distance or time of the X-Y table is minimized. In the TSP model, a decision variable, x_{ij} , is normally used to indicate that component i is placed immediately before component j if $x_{ij} = 1$. However, sub-tours may be formed and thus the bulky sub-tour elimination constraints are essential in the model. Here, x_{ip} instead of x_{ij} is used as the decision variable. 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, 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 appear in this situation.

For the component sequencing model, only x_{ip} is incorporated, whereas the assignment of component types to feeders (i.e., $y_{t,r}$) is predetermined. After the feeder arrangement has been generated, the objective function of the model can then be constructed as:

$$\text{Minimize } z = \max \left(\sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n C1_{ij} x_{i,p-1} x_{j,p}, \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n C2_{rs} x_{i,p+g} x_{j,p+g+1}, C3 \right) \\ \text{for } p = 1, 2, \dots, n.$$

The above objective function is to minimize the summation of all the dominating times among $C1_{ij}$, $C2_{rs}$, and $C3$ of the components. First of all, $C1_{ij}$ calculates the time used by the X-Y table for traveling from the position of component i on the PCB to the position of component j , which is placed in the p th position. As described earlier, one nozzle is placing a component on the PCB while another nozzle is picking up another component from a feeder at the same time. Here, when the p th component is being placed, the type of another component to be placed in the $(p + g + 1)$ th position is being picked up from a feeder. It is assumed that the types of the $(p + g)$ th and the $(p + g + 1)$ th components are stored in feeders r and s , respectively. So, $C2_{rs}$ calculates the time used by the feeder carrier for traveling from feeder r to feeder s . Thirdly, $C3$ is the indexing time of the turret.

After introducing T_p , which is the dominating time for assembling the p th component, and incorporating the constraint sets for determining the sequence of component placements, the component sequencing model can be constructed as:

$$\text{Minimize } z = \sum_{p=1}^n T_p \tag{1}$$

subject to

$$T_p - \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n C1_{ij} x_{i,p-1} x_{j,p} \geq 0 \quad \text{for } p = 1, 2, \dots, n. \tag{2}$$

$$T_p - \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n C2_{rs} x_{i,p+g} x_{j,p+g+1} \geq 0 \quad \text{for } p = 1, 2, \dots, n. \tag{3}$$

$$T_p - C3 \geq 0 \quad \text{for } p = 1, 2, \dots, n. \tag{4}$$

$$\sum_{i=1}^n x_{ip} = 1 \quad \text{for } p = 1, 2, \dots, n. \quad (5)$$

$$\sum_{p=1}^n x_{ip} = 1 \quad \text{for } i = 1, 2, \dots, n. \quad (6)$$

$$\text{All } x_{ip} = 0 \text{ or } 1, T_p \geq 0 \text{ and is a set of integers} \quad (M1)$$

Although the minimax type objective function is transformed into the minimization one, three constraint sets are introduced. Constraint set (2) is to calculate the traveling time of the X-Y table for assembling the p th component. Constraint set (3) is to calculate the traveling time of the feeder carrier for assembling the p th component. Constraint set (4) is the indexing time of the turret. For example, when $p = 1$, if $T_1 \geq 5$, $T_1 \geq 4$, and $T_1 \geq 3$ in constraint sets (2), (3), and (4), respectively, then T_1 will become 5 in order to satisfy all constraints. This is the idea of obtaining the value of T_p . Besides, constraint set (5) is to guarantee that exactly one component is placed in one position, while constraint set (6) is to guarantee that one position has exactly one component placed.

The assembly time of the chip shooter machine is dependent on all the $C1_{ij}$, $C2_{rs}$, and $C3$. Definitely, these three traveling times must be incorporated together in the objective function of the component sequencing model. Focusing on the traveling time of the X-Y table only cannot represent the actual situation, and therefore the TSP may not be desirable for the component sequencing problem.

3.2. A feeder arrangement model

Assuming that the sequence of component placements is predetermined, some researchers formulated the feeder arrangement problem as the QAP for assigning the component types to feeders so that the number of feeder carrier's movements or the total traveling time of the feeder carrier is minimized. As the case for the component sequencing model, the three traveling times should be considered simultaneously in the feeder arrangement model. So, the QAP may not be desirable in this case.

In the model, only $y_{i,r}$ is incorporated, whereas the sequence of component placements (i.e., x_{ip}) is known in advance. It is to determine which component type is stored in which feeder. Since it is assumed that the number of feeders available is equivalent to that of component types required, a single component type can only be assigned to a feeder. The interpretation of $y_{i,r}$ is that:

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

After x_{ip} has been known, the objective function of the feeder arrangement model is constructed as:

$$\text{Minimize } z = \max \left(C1_{ij}, \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n \sum_{t=1}^{\mu} \sum_{r=1}^{\mu} \sum_{s=1}^{\mu} C2_{rs} y_{t_i r} y_{t_j s}, C3 \right) \\ \text{for } p = 1, 2, \dots, n.$$

The above objective function calculates the total assembly time for assembling all components on a PCB. It is the summation of all the dominating times among $C1_{ij}$, $C2_{rs}$, and $C3$ of the components. Since the placement order of each component is known, the times used by the X-Y table for traveling from the position of component i to that of component j (i.e., $C1_{ij}$) can be directly obtained. It is noted that the index j in $C1_{ij}$ refers to component j to be placed in the p th position. When the position of the p th component is being moved to the placement location, the feeder holding the type of component to be placed in the $(p + g + 1)$ th position is being moved to the pick-up location. Here, the indices i and j in $C2_{rs}$ refer to components i and j , respectively. Also, component j is placed in the $(p + g + 1)$ th position, and component i is placed immediately prior to component j . If the type of component i (i.e., t_i) is stored in feeder r and the type of component j (i.e., t_j) is assigned to feeder s , $C2_{rs}$ is equivalent to the time used by the feeder carrier to travel from feeder r to feeder s .

By introducing T_p and incorporating the constraint sets for determining the assignment of component types to feeders, the feeder arrangement model can be formulated as:

$$\text{Minimize } z = \sum_{p=1}^n T_p \quad (7)$$

subject to

$$T_p - C1_{ij} \geq 0 \quad \text{for } p = 1, 2, \dots, n. \quad (8)$$

$$T_p - \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n \sum_{t=1}^{\mu} \sum_{r=1}^{\mu} \sum_{s=1}^{\mu} C2_{rs} y_{t_i r} y_{t_j s} \geq 0 \quad \text{for } p = 1, 2, \dots, n. \quad (9)$$

$$T_p - C3 \geq 0 \quad \text{for } p = 1, 2, \dots, n. \quad (10)$$

$$\sum_{t=1}^{\mu} y_{tr} = 1 \quad \text{for } r = 1, 2, \dots, \mu. \quad (11)$$

$$\sum_{r=1}^{\mu} y_{tr} = 1 \quad \text{for } t = 1, 2, \dots, \mu. \quad (12)$$

All $y_{tr} = 0$ or 1 , $T_p \geq 0$ and is a set of integers (M2)

Similar to M1, the minimax type objective function is transformed into a simple objective function (7) while subjecting to three constraint sets (8), (9), and (10) in M2. This transformation requires an introduction of variable T_p . Besides, two constraint sets should be incorporated to determine which feeder holds which component type. Constraint set (11) ensures that exactly one component type is stored in one feeder. Constraint set (12) ensures that exactly one feeder holds one component type.

3.3. The integrated component scheduling models

Before solving the component sequencing model (i.e., M1), it is essential to obtain the solution of the feeder arrangement problem (i.e., M2) first. On the other hand, M2 cannot be solved until the solution of M1 is known. Therefore, it is no doubt that the component sequencing and the feeder arrangement problems are inter-related. And, it is more suitable to consider the two problems simultaneously rather than separately, as many researchers did.

For the integrated problem, the objective of the complete model is to minimize the summation of all the dominating times among $C1_{ij}$, $C2_{rs}$, and $C3$ of all components on the PCB. It can be formulated as:

$$\text{Minimize } z = \max \left(\sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n C1_{ij} x_{i,p-1} x_{j,p}, \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n \sum_{t=1}^{\mu} \sum_{r=1}^{\mu} \sum_{s=1}^{\mu} C2_{rs} x_{i,p+g} x_{j,p+g+1} y_{t,r} y_{t,s}, C3 \right) \\ \text{for } p = 1, 2, \dots, n.$$

When the p th component is being moved to the placement location, the feeder holding the type of $(p + g + 1)$ th component is being moved to the pick-up location, as shown in Fig. 1. The pick-up and the placement operations take place after the end of the movements.

After adding the decision variable T_p , and the constraints for determining the component sequencing and the feeder arrangement, the complete model for the integrated problem in the form of minimization is formulated as:

$$\text{Minimize } z = \sum_{p=1}^n T_p \quad (13)$$

subject to

$$T_p - \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n C1_{ij} x_{i,p-1} x_{j,p} \geq 0 \quad \text{for } p = 1, 2, \dots, n. \quad (14)$$

$$T_p - \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n \sum_{t=1}^{\mu} \sum_{r=1}^{\mu} \sum_{s=1}^{\mu} C2_{rs} x_{i,p+g} x_{j,p+g+1} y_{t,r} y_{t,s} \geq 0$$

for $p = 1, 2, \dots, n$. (15)

$$T_p - C3 \geq 0$$

for $p = 1, 2, \dots, n$. (16)

$$\sum_{i=1}^n x_{ip} = 1$$

for $p = 1, 2, \dots, n$. (17)

$$\sum_{p=1}^n x_{ip} = 1$$

for $i = 1, 2, \dots, n$. (18)

$$\sum_{t=1}^{\mu} y_{tr} = 1$$

for $r = 1, 2, \dots, \mu$. (19)

$$\sum_{r=1}^{\mu} y_{tr} = 1$$

for $t = 1, 2, \dots, \mu$. (20)

$$\text{All } x_{ip} \text{ and } y_{tr} = 0 \text{ or } 1, T_p \geq 0 \text{ and is a set of integers} \quad (\text{M3})$$

M3 is a pure integer nonlinear programming model. The interpretation of the objective function and the constraint sets can be found in M1 and M2. M3 has $(n^2 + \mu^2)$ binary variables, n integer variables, and $(5n + 2\mu)$ constraints. In each of the constraint sets (14) and (15), the terms are $n(n - 1)$ and $n\mu^2(n - 1)$, respectively. For a realistically sized problem of 100 components and 10 component types, M3 has 10,100 binary variables, 100 integer variables, and 520 constraints. In addition, M3 has 9,900 and 990,000 terms in each of the constraint sets (14) and (15), respectively!

Due to the fact that nonlinear programming model is very difficult to solve, it is desirable to reformulate M3 into linear programming, if possible. Since the nonlinear terms in the constraint sets (14) and (15) are in the form of products of binary variables, M3 can be reformulated as a linear one by implementing the following two steps: (a) Replace the product term xy by a binary variable w ; (b) Impose the logical condition: $w = 1$ if and only if $x = 1$ and $y = 1$ by means of the extra constraints: $w \leq x$, $w \leq y$, and $w \geq x + y - 1$.

In the constraint set (14) of M3, the nonlinear term is in the form of products of two binary variables. So, it can be rewritten as a linear constraint by introducing an extra binary variable w_{ijp} and three extra constraint sets. The interpretation of the decision variable w_{ijp} is:

$$w_{ijp} = \begin{cases} 1 & \text{if component } j \text{ is placed just after component } i \text{ and} \\ & \text{component } j \text{ is placed in the } p\text{th position,} \\ 0 & \text{otherwise.} \end{cases}$$

On the other hand, in the constraint set (15) of M3, the nonlinear term is in the form of products of four binary variables. So, the steps for converting it into linear type need to be modified in this case. The major difference is that five instead of three extra constraints are introduced. Similar to that for the constraint set (14), a decision variable $w_{ij(p+g+1)rs}$ is introduced. The interpretation of the decision variable $w_{ij(p+g+1)rs}$ is:

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

The mathematical model in the form of linear function can be formulated as follows:

$$\text{Minimize } z = \sum_{p=1}^n T_p \quad (21)$$

subject to

$$T_p - \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n C1_{ij} w_{ijp} \geq 0 \quad \text{for } p = 1, 2, \dots, n. \quad (22)$$

$$T_p - \sum_{i=1}^n \sum_{\substack{j=1 \\ j \neq i}}^n \sum_{r=1}^{\mu} \sum_{s=1}^{\mu} C2_{rs} w_{ij(p+g+1)rs} \geq 0 \quad \text{for } p = 1, 2, \dots, n. \quad (23)$$

$$T_p - C3 \geq 0 \quad \text{for } p = 1, 2, \dots, n. \quad (24)$$

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

$$\sum_{p=1}^n x_{ip} = 1 \quad \text{for } i = 1, 2, \dots, n. \quad (26)$$

$$\sum_{t=1}^{\mu} y_{tr} = 1 \quad \text{for } r = 1, 2, \dots, \mu. \quad (27)$$

$$\sum_{r=1}^{\mu} y_{tr} = 1 \quad \text{for } t = 1, 2, \dots, \mu. \quad (28)$$

$$w_{ijp} \leq x_{i,p-1} \quad \text{for } i, j, p = 1, 2, \dots, n; i \neq j. \quad (29)$$

$$w_{ijp} \leq x_{jp} \quad \text{for } i, j, p = 1, 2, \dots, n; i \neq j. \quad (30)$$

$$w_{ijp} \geq x_{i,p-1} + x_{jp} - 1 \quad \text{for } i, j, p = 1, 2, \dots, n; i \neq j. \quad (31)$$

$$w_{ij(p+g+1)rs} \leq x_{i,p+g} \quad \text{for } i, j, p = 1, 2, \dots, n; i \neq j; \text{ for } r, s = 1, 2, \dots, \mu. \quad (32)$$

$$w_{ij(p+g+1)rs} \leq x_{j,p+g+1} \quad \text{for } i, j, p = 1, 2, \dots, n; i \neq j; \text{ for } r, s = 1, 2, \dots, \mu. \quad (33)$$

$$w_{ij(p+g+1)rs} \leq y_{t,r} \quad \text{for } i, j, p = 1, 2, \dots, n; i \neq j; \text{ for } r, s = 1, 2, \dots, \mu. \quad (34)$$

$$w_{ij(p+g+1)rs} \leq y_{t_{js}} \quad \text{for } i, j, p = 1, 2, \dots, n; i \neq j; \text{ for } r, s = 1, 2, \dots, \mu. \quad (35)$$

$$w_{ij(p+g+1)rs} \geq x_{i,p+g} + x_{j,p+g+1} + y_{t_{ir}} + y_{t_{js}} - 3 \\ \text{for } i, j, p = 1, 2, \dots, n; i \neq j; \text{ for } r, s = 1, 2, \dots, \mu. \quad (36)$$

$$\text{All } x_{ip}, y_{tr}, w_{ijp}, \text{ and } w_{ij(p+g+1)rs} = 0 \text{ or } 1, T_p \geq 0 \text{ and is a set of integers} \quad (\text{M4})$$

In M4, constraint sets (22), and (29) to (31) are the linear expression of the constraint set (14), whereas constraint sets (23), and (32) to (36) are the linear expression of the constraint set (15). Although the model becomes linear, both numbers of variables and constraints in M4 increase greatly. For the number of binary variables, $n^2(n-1)$ of w_{ijp} and $n^2\mu^2(n-1)$ of $w_{ij(p+g+1)rs}$ are introduced in M4 besides $(n^2 + \mu^2)$ binary variables and n integer variables. For the number of constraints, besides $(5n + 2\mu)$ constraints, M4 has $[3n^2(n-1)] + [5n^2\mu^2(n-1)]$ constraints more in which there are $[3n^2(n-1)]$ constraints for constraint sets (29) to (31) and there are $[5n^2\mu^2(n-1)]$ constraints for constraint sets (32) to (36). For a realistically sized problem of 100 components and 10 component types, M4 has 100,000,200 variables, and 497,970,520 constraints. The numbers of variables and constraints of both M3 and M4 are summarized in Table 2.

To verify the models, two commercial packages are used. BARON is adopted to solve the pure integer nonlinear programming model or M3, whereas CPLEX is applied to solve the pure integer linear programming model or M4. By these two commercial packages, M3 and M4 are tested by several small examples, and they can both generate the same optimal solutions to the same problems. The computational time spent on solving the models on a 2.26GHz computer is shown in Table 3. It is found that M3 is more desirable than M4 in terms of the amount of computational time spent. Because of this, an integrated problem with 10 components and 6 component types is formulated as M3. The data of the problem is listed in Table 4. The optimal sequence of component placements is (10, 9, 8, 7, 6, 1, 2, 3, 4, 5). Besides, the optimal feeder arrangement is that component types 2, 4, 3, 1, 5, and 6 are stored in feeders 1, 2, 3, 4, 5, and 6, respectively. The minimum assembly time is 4 seconds. Table 5 illustrates the calculation of the three traveling, and shows how T_p is obtained. According to the above example, T_p is a positive variable. It can be converted into an integer variable by multiplying it with a constant value (e.g., 10000).

Although the models can be solved to global optimality, it is not an efficient method due to the fact that the computational time grows exponentially with the problem size. As a result, it is necessary to develop a heuristic method to solve the integrated problem for the chip shooter machine efficiently. However, a heuristic may not achieve the optimal solution.

4. A hybrid genetic algorithm

Due to the complexity of the integrated problem, a heuristic method is desirable to be adopted. The success of GAs in solving a wide variety of complex optimization problems (Goldberg, 1989; Gen and Cheng, 1997) and the advantages of GAs such as simplicity, easy operation, and great flexibility have prompted us to apply GA. Nevertheless, a simple GA may not perform well in this situation because the component sequencing and the feeder arrangement problems are considered simultaneously. Actually, the individual problems are already very hard to solve (Crama et al., 1997). The GA adopted here is therefore hybridized with several heuristics in order to improve the solution further. Recently, hybrid genetic algorithms have been widely applied to tackle hard optimization problems (Huang, 2007; Farahani and Elahipanah, 2008; ElMekkawy and Liu, 2009; Lova et al., 2009).

The flowchart of the hybrid GA or 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 have been set up, the HGA generates an initial population in which the first links are generated from the nearest neighbor heuristic (NNH) while the second links are generated randomly. During this initialization step, each chromosome is improved as follows: the 2-opt local search heuristic is applied to the second link while the iterated swap procedure (ISP) (Fig. 4), is performed on the first link. The principle of the 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 has been described thoroughly in Section 3. The roulette wheel selection operation 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 second link is improved by the 2-opt local search heuristic while the first link is improved by the ISP. The fitness of the offspring will be measured and may become a member of the 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.

5. Performance analysis

In this section, the performance of the HGA is evaluated by comparing it to the optimal solutions of several problems mentioned in Section 3, which were obtained using either BARON or CPLEX. Then, the comparison between the HGA and the GAs developed by Leu et al. (1993) as well as Ong and Tan (2002) is carried out. In addition, the effect of the population size on the effectiveness of the HGA is studied.

5.1. Comparison to optimal solutions

The same problems as mentioned in Section 3 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 5 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 14 hours when compared with BARON, and saves about 14 days when compared with CPLEX.

5.2. Comparison to other approaches

The performance of the HGA is evaluated using the PCB example in Leu et al. (1993). The HGA parameters are: *population size* = 25, *iteration number* = 1000, *crossover rate* = 0.4, and *mutation rate* = 0.2. According to Table 6, it is found that the performance of the HGA is superior to that of the GAs used in Leu et al. (1993) and Ong and Tan (2002) in three aspects. Firstly, the best chromosome (30 seconds) in the initial population obtained by the HGA is better than that of the simple GAs (both more than 60 seconds). Secondly, the HGA can obtain a better solution with fewer iterations, 323 vs. 1,750 or 5,000. Finally and the most importantly, the HGA obtained a better solution than the previous methods, 26 seconds vs. 51.5 seconds or 26.9 seconds. This improvement of just 0.9 second when compared with Ong and Tan (2002) is significant in the PCB assembly industry. Since the component placement is the bottleneck of the assembly line as mentioned in Section 1, a minor reduction in the cycle time will save a significant production time. For example, to produce 100,000 boards, a reduction of 0.9 second in the cycle time will save 1,500 minutes or 25 working hours. Therefore, the productivity of a PCB manufacturing company can be enhanced if our HGA is adopted.

5.3. Effect of population size

Population size plays a vital role since it determines the effectiveness of the HGA. The larger the population size, the higher the chance of finding the optimal solution or a better

solution. In order to identify the effect of the population size, the HGA program is run with a 100-component problem and a 200-component problem each of which using three different population sizes: 5, 25, and 50. Fig. 8 and Fig. 9 show the performance of the HGA for the 100-component and 200-component problems, respectively. It can be seen that the curves representing the population size of 50 are the lowest. This shows that the HGA with a larger population size can obtain a better final solution.

6. Conclusions

This paper considered the component sequencing and the feeder arrangement problems simultaneously for the chip shooter machine, with the objective of minimizing the total assembly time. Two individual mathematical models were formulated for the individual component sequencing and feeder arrangement problems first. It was noticed that the problems are inter-related. One cannot be solved unless the solution of the other one is obtained beforehand. Due to their inseparable relationship, a pure integer nonlinear programming model and a pure integer linear programming model for the integrated problem were formulated. The nonlinear and the linear programming models were verified using the commercial packages, and both of them generated the same optimal solutions to the same problems. Different types of models took different times for computation. In terms of the amount of computational time spent for solving the model to global optimality, the nonlinear type is preferred.

Although the optimal solution could be found using the commercial packages, it was found that the computational time grows exponentially with the problem size. As a result, a HGA was adopted to solve the integrated problem. It was found that the performance of the HGA is desirable not only because it can reach the global optimum of several problems with small sizes quickly, but also it can generate better solutions than the others approaches in terms of the total assembly time. Furthermore, the HGA with a larger population size can obtain a better final solution.

Acknowledgements

The work described in this paper has been supported by the Research Grants Council of Hong Kong, China (Project no.: PolyU 5259/04E).

References

- Altinkemer, K., Kazaz, B., Köksalan, M., Moskowitz, H., 2000. Optimization of printed circuit board manufacturing: integrated modeling and algorithms. *European Journal of Operational Research* 124, 409–421.
- Bard, J.F., Clayton, R.W., Feo, T.A., 1994. Machine setup and component placement in printed circuit board assembly. *International Journal of Flexible Manufacturing Systems* 6, 5–31.
- Chan, D., Mercier, D., 1989. IC insertion: an application of the travelling salesman problem. *International Journal of Production Research* 27, 1837–1841.
- Crama, Y., Flippo, O.E., Klundert, J.V.D., Spieksma, F.C.R., 1997. The assembly of printed circuit boards: a case with multiple machines and multiple board types. *European Journal of Operational Research* 98, 457–472.
- Dikos, A., Nelson, P.C., Tirpak, T.M., Wang, W., 1997. Optimization of high-mix printed circuit card assembly using genetic algorithms. *Annals of Operations Research* 75, 303–324.
- Ellis, K.P., Vittes, F.J., Kobza, J.E., 2001. Optimizing the performance of a surface mount placement machine. *IEEE Transactions on Electronics Packaging Manufacturing* 24, 160–170.
- ElMekkawy, T.Y., Liu, S., 2009. A new memetic algorithm for optimizing the partitioning problem of tandem AGV systems. *International Journal of Production Economics* 118, 508–520.
- Farahani, R.Z., Elahipanah, M., 2008. A genetic algorithm to optimize the total cost and service level for just-in-time distribution in a supply chain. *International Journal of Production Economics* 111, 229–243.
- Gen, M., Cheng, R., 1997. *Genetic Algorithms and Engineering Design*. Wiley, New York.
- Goldberg, D.E., 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, New York.
- Huang, X., 2007. Optimal project selection with random fuzzy parameters. *International Journal of Production Economics* 106, 513–522.
- Klomp, C., Klundert, J.V.D., Spieksma, F.C.R., Voogt, S., 2000. The feeder rack assignment problem in PCB assembly: a case study. *International Journal of Production Economics* 64, 399–407.
- Leu, M.C., Wong, H., Ji, Z., 1993. Planning of component placement/insertion sequence and feeder setup in PCB assembly using genetic algorithm. *Journal of Electronic Packaging* 115, 424–432.

- Lova, A., Tormos, P., Cervantes, M., Barber, F., 2009. An efficient hybrid genetic algorithm for scheduling projects with resource constraints and multiple execution modes. *International Journal of Production Economics* 117, 302–316.
- Moyer, L.K., Gupta, S.M., 1996a. SMT feeder slot assignment for predetermined component placement paths. *Journal of Electronics Manufacturing* 6, 173–192.
- Moyer, L.K., Gupta, S.M., 1996b. Simultaneous component sequencing and feeder assignment for high speed chip shooter machines. *Journal of Electronics Manufacturing* 6, 271–305.
- Moyer, L.K., Gupta, S.M., 1997. An efficient assembly sequencing heuristic for printed circuit board configurations. *Journal of Electronics Manufacturing* 7, 143–160.
- Ong, N.S., Tan, W.C., 2002. Sequence placement planning for high-speed PCB assembly machine. *Integrated Manufacturing Systems* 13, 35–46.
- Sohn, J., Park, S., 1996. Efficient operation of a surface mounting machine with a multihead turret. *International Journal of Production Research* 34, 1131–1143.
- Souza, R.D., Wu, L.J., 1995. Intelligent optimization of component onsertion in multi-head concurrent operation PCBA machines. *Journal of Intelligent Manufacturing* 6, 235–243.
- Wilhelm, W.E., Tarmy, P.K., 2003. Circuit card assembly on tandem turret-type placement machines. *IIE Transactions* 35, 627–645.
- Yeo, S.H., Low, C.W., Yong, K.H., 1996. A rule-based frame system for concurrent assembly machines. *International Journal of Advanced Manufacturing Technology* 12, 370–376.

Fig. 1. The schematic diagram of the chip shooter machine.

Fig. 2. The flowchart of the HGA.

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

Select 2 genes randomly

Parent:

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Offspring 1:

1	2	8	4	5	6	7	3	9	10
---	---	---	---	---	---	---	---	---	----

Swap the neighbors of the 2 genes to form 4 more offspring

Offspring 2:

1	8	2	4	5	6	7	3	9	10
---	---	---	---	---	---	---	---	---	----

Offspring 3:

1	2	4	8	5	6	7	3	9	10
---	---	---	---	---	---	---	---	---	----

Offspring 4:

1	2	8	4	5	6	3	7	9	10
---	---	---	---	---	---	---	---	---	----

Offspring 5:

1	2	8	4	5	6	7	9	3	10
---	---	---	---	---	---	---	---	---	----

Fig. 4. The iterated swap procedure.

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:

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Generate neighbors for all possible permutations of the selected genes, and all neighbors generated are regarded as the offspring.

Offspring 1:

1	2	3	4	5	8	7	6	9	10
---	---	---	---	---	---	---	---	---	----

Offspring 2:

1	2	6	4	5	3	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Offspring 3:

1	2	6	4	5	8	7	3	9	10
---	---	---	---	---	---	---	---	---	----

Offspring 4:

1	2	8	4	5	3	7	6	9	10
---	---	---	---	---	---	---	---	---	----

Offspring 5:

1	2	8	4	5	6	7	3	9	10
---	---	---	---	---	---	---	---	---	----

Fig. 6. The heuristic mutation operator.

Fig. 7. The inversion mutation operator.

Fig. 8. The effect of population size (100-component problem).

Fig. 9. The effect of population size (200-component problem).

Table 1

Notation

Indices:

i, j : components ($i, j = 1, 2, \dots, n$).

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

r, s : feeders ($r, s = 1, 2, \dots, \mu$).

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

Traveling times:

$C1_{ij}$: traveling time of the X-Y table.

$C2_{rs}$: traveling time of the feeder carrier.

$C3$: indexing time of the turret.

T_p : the longest traveling time among three for assembling the p th component.

Feeder constraint:

g : number of components between the pick-up and the placement locations.

Decision variables:

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

$y_{i,r} = 1$ if component type t of component i is stored in feeder r ; 0 otherwise.

Table 2

Numbers of variables and constraints in M3 and M4

	M3	M4
<i>No. of variables</i>	$n^2 + \mu^2 + n$	$(n^2 + \mu^2 + n) + [n^2(n - 1)] + [n^2\mu^2(n - 1)]$
<i>No. of constraints</i>	$5n + 2\mu$	$(5n + 2\mu) + [3n^2(n - 1)] + [5n^2\mu^2(n - 1)]$

Table 3

Computational times spent by BARON and CPLEX for solving M3 and M4

Numbers of components and types	Optimal solution CPU time (hh:mm:ss) by BARON for M3	Optimal solution CPU time (hh:mm:ss) by CPLEX for M4
4×4	00:00:01	00:00:01
5×5	00:00:14	00:00:20
6×6	00:02:27	00:07:19
7×7	00:47:20	04:37:39
8×8	14:30:25	328:07:51

Table 4

Data of the integrated problem with 10 components and 6 component types

Components	Types	Coordinates (mm)		Feeders	Coordinates (mm)	
		x	y		x	y
1	6	100	80	1	100	300
2	5	120	80	2	115	300
3	1	140	80	3	130	300
4	3	160	80	4	145	300
5	4	180	80	5	160	300
6	5	100	40	6	175	300
7	1	120	40			
8	3	140	40			
9	4	160	40			
10	2	180	40			

Remarks: The number of components between the pick-up and the placement locations in the turret, $g = 2$; The table's speed in both x and y directions, $V_x = V_y = 60$ mm/s; The linear speed of the feeder carrier, $V_f = 60$ mm/s; The indexing time of the head, $C3 = 0.25$ second per index.

Table 5

Calculation of the three traveling times in the integrated problem

	$C1_{ij}$	$C2_{rs}$	$C3$	T_p
$p = 1$	0.6667 ($c_5 \rightarrow c_{10}$)	0.25 ($f_3 \rightarrow f_4$)	0.25	0.6667
$p = 2$	0.3333 ($c_{10} \rightarrow c_9$)	0.25 ($f_4 \rightarrow f_5$)	0.25	0.3333
$p = 3$	0.3333 ($c_9 \rightarrow c_8$)	0.25 ($f_5 \rightarrow f_6$)	0.25	0.3333
$p = 4$	0.3333 ($c_8 \rightarrow c_7$)	0.25 ($f_6 \rightarrow f_5$)	0.25	0.3333
$p = 5$	0.3333 ($c_7 \rightarrow c_6$)	0.25 ($f_5 \rightarrow f_4$)	0.25	0.3333
$p = 6$	0.6667 ($c_6 \rightarrow c_1$)	0.25 ($f_4 \rightarrow f_3$)	0.25	0.6667
$p = 7$	0.3333 ($c_1 \rightarrow c_2$)	0.25 ($f_3 \rightarrow f_2$)	0.25	0.3333
$p = 8$	0.3333 ($c_2 \rightarrow c_3$)	0.25 ($f_2 \rightarrow f_1$)	0.25	0.3333
$p = 9$	0.3333 ($c_3 \rightarrow c_4$)	0.25 ($f_1 \rightarrow f_2$)	0.25	0.3333
$p = 10$	0.3333 ($c_4 \rightarrow c_5$)	0.25 ($f_2 \rightarrow f_3$)	0.25	0.3333
Total assembly time =				4.0000

Table 6

A comparison of the experimental results

	Leu et al. (1993)	Ong and Tan (2002)	HGA
<i>Best one in the initial solution (s)</i>	70	About 60	30
<i>Population size</i>	100	10	25
<i>Iteration number</i>	About 1,750	5,000	323
<i>Final best solution (s)</i>	About 51.5	26.9	26

Minerva Access is the Institutional Repository of The University of Melbourne

Author/s:

Ho, W;Ji, P

Title:

Integrated component scheduling models for chip shooter machines

Date:

2010-01-01

Citation:

Ho, W. & Ji, P. (2010). Integrated component scheduling models for chip shooter machines. INTERNATIONAL JOURNAL OF PRODUCTION ECONOMICS, 123 (1), pp.31-41. <https://doi.org/10.1016/j.ijpe.2009.05.029>.

Persistent Link:

<http://hdl.handle.net/11343/118667>