(Research Paper)

PCB assembly line assignment: a genetic algorithm approach

W. Ho and P. Ji*

Department of Industrial and Systems Engineering

The Hong Kong Polytechnic University

Hung Hom, Kowloon, Hong Kong

Email: mfpji@inet.polyu.edu.hk

Abstract

This paper studies a production planning problem for printed circuit board (PCB) assembly. A PCB assembly company may have a number of assembly lines for production of several product types in large volume. Pure integer linear programming models are formulated for assigning the product types to assembly lines, which is the line assignment problem, with the objective of minimizing the total production cost. In our approach, unrealistic assignment, which was suffered by previous researchers, is avoided by incorporating several constraints into the model. In this paper, a genetic algorithm is developed to solve the line assignment problem. The procedure of the genetic algorithm to the problem, and a numerical example for illustrating the models are provided. It is also proved that the algorithm is effective and efficient in dealing with the problem.

Keywords

Assembly, Modeling, Printed circuit boards, Genetic algorithm.

* Corresponding author

Introduction

Printed circuit boards (PCBs) are extensively used in a variety of products today. These products include, for example, computers, mobile phones, and audiovisual equipment. The wide applicability of PCB has motivated researchers or process planners to concentrate on the production planning in order to reduce the production cost and remain competitive. In order to achieve these goals, several PCB assembly problems need to be considered. Ellis *et al.* summarized that there are seven problems related to the PCB production planning (2001). They involve the line assignment problem, the machine grouping problem, the PCB grouping problem, the PCB sequencing problem, the component allocation problem, the feeder arrangement problem, and the component sequencing problem.

The first problem or the line assignment problem is studied in this paper. A PCB assembly company may have several assembly lines either in the pin-through-hole technology or the surface mount technology to cope with customers' orders. Product types must be assigned to appropriate lines such that the production is on time. Hillier and Brandeau (1998) formulated an integer linear programming model for assigning boards and components to the machines and the manual process so as to minimize the total cost for PCB and components setup. An optimal solution technique was developed for the single-machine case and for the multi-machine case where boards were not allowed to be setup on more than one process. Also, a heuristic approach was developed to obtain a near-optimal solution. Balakrishnan and Vanderbeck (1999) developed an integer linear programming model that minimized the setup costs of the placement machines while ensuring that the total processing workload on each line did not exceed a predetermined limit. The model was to assign product families to parallel surface mount assembly lines. An optimization-based method incorporated with column generation, heuristics, and lower bound procedures, was adopted to obtain a near-optimal solution. Hillier and Brandeau (2001) formulated an integer linear programming model for

assigning boards and components to the machines and the manual process in order to minimize the production cost and to balance the machine workloads. The machine capacity was also taken into consideration. A heuristic called the cost minimizing workload balancing was developed to generate the upper bounds. The branch-and-bound method was used to find the optimal solutions to small and medium-sized problems. Ellis and Bhoja (2002) formulated the line assignment problem as a mixed integer linear programming model, with the objective of minimizing the total assembly time including the setup time and the processing time for each of the board types. The problem was then solved using problem decomposition along with the branch-and-bound algorithm. However, the above researchers did not mention and consider a special case that a few board types, say less than 10, may be assigned to a particular assembly line. It is unreasonable to set up a line for producing one or few boards. Thus, the line assignment problem has not been thoroughly studied yet.

Due to the complexity of the PCB assembly problems, heuristic approaches are often adopted to solve the problems (Ong and Khoo, 1999). Although there are many alternative heuristic methods to genetic algorithms (GAs), there is still no clear guidance available regarding the optimal choice for a particular problem. On the other hand, GAs have been applied successfully in a wide variety of optimization problems: the traveling salesman problem, the quadratic assignment problem, and the minimum spanning tree problem (Gen and Cheng, 1997). However, it is found that none of the researchers has applied a GA to solve the line assignment problem. So, a GA is developed to solve the problem in this paper.

Mathematical models

Suppose that m assembly lines are available for the production of n types of products, in which the production volume is high. Here, production means that each product is processed in a single line instead of by a specified sequence of lines. Furthermore, a product can be produced

in any line. When line i is assigned to produce product j , it requires $a_{ij} (> 0)$ hours and costs $c_{ij} (> 0)$ dollars to make one unit j . Besides, line i has a maximum of t_i hours available for production while product j has a volume requirement of s_j . The problem here is to determine the quantity x_{ij} of product j to be produced on line i in order to minimize the total production cost. A pure integer linear programming model can be formulated as follows:

$$\text{Minimize } z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} \quad (1)$$

subject to

$$\sum_{j=1}^n a_{ij} x_{ij} \leq t_i \quad i = 1, 2, \dots, m \quad (2)$$

$$\sum_{i=1}^m x_{ij} = s_j \quad j = 1, 2, \dots, n \quad (3)$$

$$x_{ij} \geq 0 \text{ and is a set of integers} \quad (4) \quad (\text{IP1})$$

The objective function (1) is to minimize the total production cost. Constraint set (2) is due to the limited available time, which means each line must be operated within the fixed time period. Constraint set (3) is from the product volume requirement. Constraint set (4) is obvious because the quantity of a board type assigned to an assembly line must be integer. Since model IP1 consists solely of integer variables, it is regarded as the pure integer linear programming model. Table I below shows a line assignment problem tableau, which has four lines and five types of products. The northwest corner in cell (i, j) indicates the unit time a_{ij} , while the northeast corner represents the unit cost c_{ij} . For example, in cell $(1, 1)$, the unit production time is 5 and the unit cost is 3.

take in Table I

However, the optimal assignment of board types to assembly lines may not be so reasonable in the real cases. For example, if one or few boards are assigned to a particular

assembly line, it is not practical to set up the line for processing only few boards indeed. Thus, model IP1 needs some modifications for avoiding such impractical assignment. In order to achieve this goal, three additional binary variables are introduced. The interpretations of the variables are as follows:

$$u_{ij} = \begin{cases} 1 & \text{if quantity of board type } j \text{ assigned to line } i \text{ less than } q, \\ 0 & \text{otherwise.} \end{cases}$$

$$v_{ij} = \begin{cases} 1 & \text{if quantity of board type } j \text{ assigned to line } i \text{ more than } 0, \\ 0 & \text{otherwise.} \end{cases}$$

$$w_{ij} = \begin{cases} 1 & \text{if both } u_{ij} \text{ and } v_{ij} \text{ equal 1,} \\ 0 & \text{otherwise.} \end{cases}$$

Here, for example, the assignment is regarded as infeasible if the quantity of a board type assigned to a line is less than q , which represents the minimum quantity of assignment, but more than zero (i.e., $0 < x_{ij} < q$). In this case, penalty p will be included. Also, it is noted that M denotes a constant coefficient representing a known upper bound for x_{ij} . The mathematical model for avoiding impractical assignment can be formulated as follows:

$$\text{Minimize } z = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} + p \sum_{i=1}^m \sum_{j=1}^n w_{ij} \quad (5)$$

subject to

$$\sum_{j=1}^n a_{ij} x_{ij} \leq t_i \quad i = 1, 2, \dots, m \quad (2)$$

$$\sum_{i=1}^m x_{ij} = s_j \quad j = 1, 2, \dots, n \quad (3)$$

$$x_{ij} + M u_{ij} \geq q \quad i = 1, 2, \dots, m; j = 1, 2, \dots, n \quad (6)$$

$$x_{ij} - M v_{ij} \leq 0 \quad i = 1, 2, \dots, m; j = 1, 2, \dots, n \quad (7)$$

$$w_{ij} \geq u_{ij} + v_{ij} - 1 \quad i = 1, 2, \dots, m; j = 1, 2, \dots, n \quad (8)$$

$$x_{ij} \geq 0 \text{ and is a set of integers; } u_{ij}, v_{ij}, \text{ and } w_{ij} = 0 \text{ or } 1 \quad (9) \quad (\text{IP2})$$

The objective function of model IP2 is different from that of model IP1. The objective function (5) is to minimize the combination of total cost as well as penalty incurred by impractical assignment, which means that the amount of a board type assigned to an assembly line is lower than the minimum assignment quantity. Besides, three additional constraint sets are included in model IP2. They are used to reflect the fact that whether the assignment is practical or not. For instance, if $0 < x_{ij} < q$, then both u_{ij} in constraint set (6) and v_{ij} in constraint set (7) will be equal to 1. In this case, w_{ij} will be equal to 1 also, and penalty p will be incurred in the objective function.

Model IP2 can be viewed as a pure integer linear programming model too, since it consists of both integer variables (i.e., x_{ij}), as well as binary variables (i.e., u_{ij} , v_{ij} , and w_{ij}). Obviously, the computing complexity of model IP2 is much greater than that of model IP1. Both numbers of decision variables and constraints are much greater in model IP2. Besides mn number of x_{ij} , model IP2 has $3mn$ number of binary variables more. At the same time, model IP2 has $3mn$ number of constraints more. For a realistically sized problem of 10 assembly lines and 100 board types, IP2 has 1,000 integer variables, 3,000 binary variables, and 3,110 constraints!

A genetic algorithm

In order to solve the line assignment problem efficiently, a genetic algorithm (GA) is applied. The overall structure of the GA for the problem is shown in Figure 1. The GA starts with an initial set of random solutions called population. Each solution in the population is called a chromosome or parent. During each generation, the parent is evaluated using some measures of fitness. The fitter the parent, the higher the probabilities of being selected to perform a genetic operation (either crossover or mutation). After new chromosomes, or offspring, are generated, the fitness of the offspring will be measured and then compared with the parents.

An offspring will become a member of population once 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 GA will not stop unless the predetermined number of iterations is conducted. Finally, the GA converges to the best chromosome to the problem.

take in Figure 1

The mechanisms of the GA are described in detail in the following subsections. Here, a matrix is used to represent a chromosome in the GA, as shown in Figure 2.

take in Figure 2

Initialization

The following initialization procedure is used to generate a feasible chromosome for the line assignment problem and it will fulfil the constraint sets (2) and (3) in model IP2.

- Step 1: Select a random number k from set π , $\pi = \{1, 2, \dots, mn\}$.
- Step 2: Calculate the corresponding row and column numbers i and j by: $i = \lfloor (k-1) / n + 1 \rfloor$ and $j = (k-1) \bmod n + 1$.
- Step 3: Assign the available amount of units to x_{ij} : $x_{ij} = \min(\frac{t_i}{a_{ij}}, s_j)$.
- Step 4: Update t_i and s_j : $t_i = t_i - x_{ij} \times a_{ij}$ and $s_j = s_j - x_{ij}$. And, delete k from π .
- Step 5: Repeat Step 1 to Step 4 until π becomes empty.

The population size is denoted as $psize$ and is predetermined by the GA user. The above procedure will be repeated $psize$ times to generate $psize$ chromosomes, that is, $h = 1, 2, \dots, psize$ in this initialization process.

Evaluation

In a GA, both parents and offspring must be evaluated by some measures of fitness. It is

because the fitness function is necessary for the subsequent procedures including the selection operation and the culling operation, which rejects the worst chromosomes in order to keep the population size constant. For the line assignment problem, the objective function of model IP2 is used for the evaluation purpose. Let $eval(X_h)$ be the fitness function for chromosome X_h in the line assignment problem, then

$$eval(X_h) = \sum_{i=1}^m \sum_{j=1}^n c_{ij} x_{ij} + p \sum_{i=1}^m \sum_{j=1}^n w_{ij}$$

Selection

In the GA for the line assignment problem, the roulette wheel approach (Goldberg, 1989) is adopted in order to choose some chromosomes to undergo genetic (crossover and mutation) operations probabilistically, instead of deterministically. For example, although one chromosome has the highest fitness, there is no guarantee it will be selected. The only thing that is certain is that on the average a chromosome will be chosen with the probability proportional to its fitness. The selection procedure is listed as follows:

Step 1: Calculate the total fitness of the population:

$$F = \sum_{h=1}^{psize} eval(X_h).$$

Step 2: Calculate the selection probability p_h for each chromosome X_h :

$$p_h = \frac{eval(X_h)}{F}, \quad h = 1, 2, \dots, psize.$$

Step 3: Calculate the cumulative probability q_h for each chromosome X_h :

$$q_h = \sum_{j=1}^h p_j, \quad h = 1, 2, \dots, psize.$$

Step 4: Generate a random number r in the range (0, 1].

Step 5: If $q_{h-1} < r \leq q_h$, then chromosome X_h is selected.

Genetic operations

The number of chromosomes selected for performing crossover and mutation operators depends on the crossover rate and the mutation rate, which are set by the GA user. Let *crossno* and *mut* denote the number of chromosomes selected to undergo the crossover and mutation, respectively, then $crossno = \text{round}(cr \times psize)$ and $mut = \text{round}(mr \times psize)$, where *cr* is the crossover rate, and *mr* is the mutation rate. Since a pair of chromosomes is required to undergo the crossover operation, the number of pairs of chromosomes, denoted as *cross*, is an integer.

$$cross = \begin{cases} \frac{crossno}{2} & \text{if } crossno \text{ is even} \\ \frac{crossno-1}{2} & \text{otherwise} \end{cases}$$

Crossover operation

Step 1: Implement the roulette wheel selection procedure twice to choose a pair of chromosomes, $X_1 = (x_{ij}^1)$ and $X_2 = (x_{ij}^2)$, from the population to perform the crossover operation.

Step 2: Create two temporary matrices, $D = (d_{ij})$ and $R = (r_{ij})$, as follows:

$$d_{ij} = \lfloor (x_{ij}^1 + x_{ij}^2) / 2 \rfloor \text{ and } r_{ij} = (x_{ij}^1 + x_{ij}^2) \bmod 2.$$

Step 3: Divide matrix R into two matrices $R_1 = (r_{ij}^1)$ and $R_2 = (r_{ij}^2)$ such that

$$R = R_1 + R_2 \text{ and}$$

$$\sum_{j=1}^n r_{ij}^1 = \sum_{j=1}^n r_{ij}^2 = \frac{1}{2} \sum_{j=1}^n r_{ij} \quad \text{for } i = 1, 2, \dots, m;$$

$$\sum_{i=1}^m r_{ij}^1 = \sum_{i=1}^m r_{ij}^2 = \frac{1}{2} \sum_{i=1}^m r_{ij} \quad \text{for } j = 1, 2, \dots, n.$$

Step 4: Produce two offspring, X_1' and X_2' , as follows:

$$X_1' = D + R_1 \text{ and } X_2' = D + R_2.$$

Mutation operation

- Step 1: Implement the selection procedure to select a chromosome.
- Step 2: Extract a sub-matrix Y from the parent matrix by randomly selecting m rows and n columns.
- Step 3: Reallocate the sub-matrix Y . Use the initialization procedure to assign new values to the sub-matrix such that all constraints are satisfied.
- Step 4: Create an offspring by replacing the appropriate elements of the parent matrix with the new elements from the reallocated sub-matrix Y .

Numerical example

A numerical example is presented to illustrate the performance of the GA. The data for the example are shown in Table I. Here, it is assumed that $p = 100$, $M = 12000$, and $q = 10$. An integer linear programming model for the numerical example can be formulated as follows:

$$\begin{aligned} \text{Minimize } z = & 3x_{11} + 6x_{12} + 6x_{13} + 7x_{14} + 4x_{15} + 6x_{21} + 5x_{22} + 6x_{23} + 15x_{24} + 8x_{25} \\ & + 5x_{31} + 3x_{32} + 10x_{33} + 6x_{34} + 7x_{35} + 4x_{41} + 4x_{42} + 7x_{43} + 8x_{44} + 9x_{45} \\ & + 100w_{11} + 100w_{12} + 100w_{13} + 100w_{14} + 100w_{15} \\ & + 100w_{21} + 100w_{22} + 100w_{23} + 100w_{24} + 100w_{25} \\ & + 100w_{31} + 100w_{32} + 100w_{33} + 100w_{34} + 100w_{35} \\ & + 100w_{41} + 100w_{42} + 100w_{43} + 100w_{44} + 100w_{45} \end{aligned}$$

subject to

$$5x_{11} + 2x_{12} + 2x_{13} + 5x_{14} + 6x_{15} \leq 48000$$

$$2x_{21} + 7x_{22} + 8x_{23} + 7x_{24} + 6x_{25} \leq 48000$$

$$4x_{31} + 2x_{32} + 2x_{33} + x_{34} + 6x_{35} \leq 48000$$

$$3x_{41} + 2x_{42} + 4x_{43} + 5x_{44} + 2x_{45} \leq 48000$$

$$x_{11} + x_{21} + x_{31} + x_{41} = 12,000$$

$$x_{12} + x_{22} + x_{32} + x_{42} = 8,000$$

$$x_{13} + x_{23} + x_{33} + x_{43} = 10,000$$

$$x_{14} + x_{24} + x_{34} + x_{44} = 8,000$$

$$x_{15} + x_{25} + x_{35} + x_{45} = 6,000$$

$$x_{11} + 12000u_{11} \geq 10; x_{12} + 12000u_{12} \geq 10; x_{13} + 12000u_{13} \geq 10; x_{14} + 12000u_{14} \geq 10;$$

$$x_{15} + 12000u_{15} \geq 10; x_{21} + 12000u_{21} \geq 10; x_{22} + 12000u_{22} \geq 10; x_{23} + 12000u_{23} \geq 10;$$

$$x_{24} + 12000u_{24} \geq 10; x_{25} + 12000u_{25} \geq 10; x_{31} + 12000u_{31} \geq 10; x_{32} + 12000u_{32} \geq 10;$$

$$x_{33} + 12000u_{33} \geq 10; x_{34} + 12000u_{34} \geq 10; x_{35} + 12000u_{35} \geq 10; x_{41} + 12000u_{41} \geq 10;$$

$$x_{42} + 12000u_{42} \geq 10; x_{43} + 12000u_{43} \geq 10; x_{44} + 12000u_{44} \geq 10; x_{45} + 12000u_{45} \geq 10$$

$$x_{11} - 12000v_{11} \leq 0; x_{12} - 12000v_{12} \leq 0; x_{13} - 12000v_{13} \leq 0; x_{14} - 12000v_{14} \leq 0;$$

$$x_{15} - 12000v_{15} \leq 0; x_{21} - 12000v_{21} \leq 0; x_{22} - 12000v_{22} \leq 0; x_{23} - 12000v_{23} \leq 0;$$

$$x_{24} - 12000v_{24} \leq 0; x_{25} - 12000v_{25} \leq 0; x_{31} - 12000v_{31} \leq 0; x_{32} - 12000v_{32} \leq 0;$$

$$x_{33} - 12000v_{33} \leq 0; x_{34} - 12000v_{34} \leq 0; x_{35} - 12000v_{35} \leq 0; x_{41} - 12000v_{41} \leq 0;$$

$$x_{42} - 12000v_{42} \leq 0; x_{43} - 12000v_{43} \leq 0; x_{44} - 12000v_{44} \leq 0; x_{45} - 12000v_{45} \leq 0$$

$$w_{11} \geq u_{11} + v_{11} - 1; w_{12} \geq u_{12} + v_{12} - 1; w_{13} \geq u_{13} + v_{13} - 1; w_{14} \geq u_{14} + v_{14} - 1;$$

$$w_{15} \geq u_{15} + v_{15} - 1; w_{21} \geq u_{21} + v_{21} - 1; w_{22} \geq u_{22} + v_{22} - 1; w_{23} \geq u_{23} + v_{23} - 1;$$

$$w_{24} \geq u_{24} + v_{24} - 1; w_{25} \geq u_{25} + v_{25} - 1; w_{31} \geq u_{31} + v_{31} - 1; w_{32} \geq u_{32} + v_{32} - 1;$$

$$w_{33} \geq u_{33} + v_{33} - 1; w_{34} \geq u_{34} + v_{34} - 1; w_{35} \geq u_{35} + v_{35} - 1; w_{41} \geq u_{41} + v_{41} - 1;$$

$$w_{42} \geq u_{42} + v_{42} - 1; w_{43} \geq u_{43} + v_{43} - 1; w_{44} \geq u_{44} + v_{44} - 1; w_{45} \geq u_{45} + v_{45} - 1$$

$$x_{ij} \geq 0 \text{ and is a set of integers; } u_{ij}, v_{ij}, \text{ and } w_{ij} = 0 \text{ or } 1 \quad (\text{IP3})$$

In the GA, the parameters are set as follows: the population size $psize = 25$, the iteration number $itno = 1,000$, the crossover rate $cr = 0.4$, and the mutation rate $mr = 0.3$. Therefore, the

number of pairs of chromosomes selected to perform the crossover operation is, $cross = 5$ while the mutation number, $mut = 8$. Figure 3 shows the minimum production cost obtained at each iteration. After 1,000 iterations, the best solution is shown in Table II with the production cost of 203,200. Although the GA cannot guarantee that the optimal solution can be found, the best solution obtained for this specific problem is optimal. The optimal integer solution can be obtained from a commercial package, like CPLEX, by solving the original integer linear programming model, IP3.

take in Figure 3 & Table II

The performance of the GA is shown in Figure 3. From the graph, it can be noticed that the objective value drops sharply at the first 90 iterations. Since the population size ($psize$) is small, only 25, the GA can only produce some not-so-good chromosomes at beginning. Later, the GA generates good offspring quickly from those highly-fit parents. This phenomenon is called rapid convergence. When the objective reaches the best solution (i.e., 203,200), the curve levels off since the best solution is already the optimal one.

Besides, the effectiveness and the efficiency of the GA are examined by solving two additional problems with medium size, and then the GA's best solutions are compared with the optimal solutions. The first problem has 10 assembly lines and 20 board types (i.e., 10×20), whereas the second one comprises of 20 assembly lines and 30 board types (i.e., 20×30). By making use of CPLEX, these two problems formulated in the form of IP2 are solved to global optimality. Then, the GA is adopted to solve the problems using the parameters: $psize = 25$, $itno = 5000$, $cr = 0.4$, and $mr = 0.3$. It is found that the GA can get the optimal solutions for both problems. Compared with the time spent on finding the optimal solutions for the problems, as shown in Table III, the GA is more efficient.

take in Table III

Conclusions

This paper studied the line assignment problem arising in a PCB manufacturing company in which the production volume is high. Two mathematical models were presented for determining the optimal assignment of several board types to multiple assembly lines with the objective of minimizing the total production cost. A tailor-made genetic algorithm was also developed to solve the line assignment problem. The procedure of the algorithm and its performance were provided in this paper. Besides, it was proved that the proposed GA could obtain the optimal solutions to some medium-sized problems more quickly than the commercial package, like CPLEX.

Acknowledgement

The authors wish to acknowledge the Hong Kong Polytechnic University for the financial support of the project (No. G-W031).

References

- Balakrishnan, A. and Vanderbeck, F. (1999), "A tactical planning model for mixed-model electronics assembly operations", *Operations Research*, Vol. 47 No. 3, pp. 395-409.
- Ellis, K.P. and Bhoja, S. (2002), "Optimization of the assignment of circuit cards to assembly lines in electronics assembly", *International Journal of Production Research*, Vol. 40 No. 11, pp. 2609-2631.
- Ellis, K.P., Vites, F.J. and Kobza, J.E. (2001), "Optimizing the performance of a surface mount placement machine", *IEEE Transactions on Electronics Packaging Manufacturing*, Vol. 24 No. 3, pp. 160-170.
- Gen, M. and Cheng, R. (1997), *Genetic Algorithms and Engineering Design*, John Wiley & Sons, New York.
- Goldberg, D.E. (1989), *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison-Wesley, New York.
- Hillier, M.S. and Brandeau, M.L. (1998), "Optimal component assignment and board grouping in printed circuit board manufacturing", *Operations Research*, Vol. 46 No. 5, pp. 675-689.
- Hillier, M.S. and Brandeau, M.L. (2001), "Cost minimization and workload balancing in printed circuit board assembly", *IIE Transactions*, Vol. 33, pp. 547-557.
- Ong, N.S. and Khoo, L.P. (1999), "Genetic algorithm approach in PCB assembly", *Integrated Manufacturing Systems*, Vol. 10, pp. 256-265.

William Ho is a PhD student at the Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University. His research area is the production planning of the printed circuit board assembly.

Ping Ji is an Associate Professor at the Department of Industrial and Systems Engineering, The Hong Kong Polytechnic University. His research areas include Production Management, Operations Research, and CAD/CAM.

Figure 1 Overall structure of GA for the line assignment problem

Figure 2 Representation of a chromosome

$$X_h = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{bmatrix}$$

Figure 3 The best production cost at each iteration

Table I A line assignment problem tableau

Product j																
Line i	1			2			3			4			5			t_i
1	5		3	2		6	2		6	5		7	6		4	48000
	x_{11}			x_{12}			x_{13}			x_{14}			x_{15}			
2	2		6	7		5	8		6	7		15	6		8	48000
	x_{21}			x_{22}			x_{23}			x_{24}			x_{25}			
3	4		5	2		3	2		10	1		6	6		7	48000
	x_{31}			x_{32}			x_{33}			x_{34}			x_{35}			
4	3		4	2		4	4		7	5		8	2		9	48000
	x_{41}			x_{42}			x_{43}			x_{44}			x_{45}			
s_j	12000			8000			10000			8000			6000			

Table II The best chromosome after 1,000 iterations

$x_{11} = 800$	$x_{12} = 0$	$x_{13} = 4,000$	$x_{14} = 0$	$x_{15} = 6,000$
$x_{21} = 0$	$x_{22} = 0$	$x_{23} = 6,000$	$x_{24} = 0$	$x_{25} = 0$
$x_{31} = 0$	$x_{32} = 8,000$	$x_{33} = 0$	$x_{34} = 8,000$	$x_{35} = 0$
$x_{41} = 11,200$	$x_{42} = 0$	$x_{43} = 0$	$x_{44} = 0$	$x_{45} = 0$
Total production cost = 203,200				

Table III Comparisons of the GA with CPLEX for the line assignment problems

Numbers of assembly lines and board types	Optimal solution CPU time (hh:mm:ss) by CPLEX	Best solution CPU time (hh:mm:ss) by GA
4×5	00:00:02	00:00:01
10×20	00:00:14	00:00:06
20×30	00:02:27	00:00:21

Minerva Access is the Institutional Repository of The University of Melbourne

Author/s:

Ho, W;Ji, P

Title:

PCB assembly line assignment: A genetic algorithm approach

Date:

2005

Citation:

Ho, W. & Ji, P. (2005). PCB assembly line assignment: A genetic algorithm approach. Journal of Manufacturing Technology Management, 16 (6), pp.682-692. <https://doi.org/10.1108/17410380510609519>.

Persistent Link:

<http://hdl.handle.net/11343/118696>