Int J Adv Manuf Technol (2000) 16:341–352

The International Journal of
Advanced
Manufacturing
Technology

A Multi-Objective Genetic Algorithm for Solving Assembly Line Balancing Problem

S. G. Ponnambalam[1], P. Aravindan[2] and G. Mogileeswar Naidu[1]

[1]Department of Mechanical Engineering, PSG College of Technology, Coimbatore 641 004 India; and [2]Regional Engineering College, Tiruchirapalli, 620 015 India

In this paper, a multi-objective genetic agorithm to solve assembly line balancing problems is proposed. The performance criteria considered are the number of workstations, the line efficiency, the smoothness index before trade and transfer, and the smoothness index after trade and transfer. The developed genetic algorithm is compared with six popular heuristic algorithms, namely, ranked positional weight, Kilbridge and Wester, Moodie and Young, Hoffmann precedence matrix, immediate update first fit, and rank and assign heuristic methods. For comparative evaluation, 20 networks are collected from open literature, and are used with five different cycle times. All the six heuristics and the genetic algorithm are coded in C++ language. It is found that the proposed genetic algorithm performs better in all the performance measures than the heuristics. However, the execution time for the GA is longer, because the GA searches for global optimal solutions with more iterations.

Keywords: Assembly line balancing; Heuristic rules; Multi-objective genetic algorithm

1. Introduction

Assembly line balancing (ALB) relates to a finite set of work elements or tasks, each having an operation processing time and a set of precedence relations, which specify the permissible orderings of the tasks. One of the problems in organising mass production is how to group work tasks to be performed on workstations so as to achieve the desired level of performance. Line balancing is an attempt to allocate equal amounts of work to the various workstations along the line. The fundamental line balancing problem is how to assign a set of tasks to an ordered set of workstations, such that the precedence relations are satisfied and some measure of performance is optimised [1].

Correspondence and offprint requests to: Dr S. G. Ponnambalam, Department of Production Engineering, Regional Engineering College, Tiruchirappalli, 620 015 India. E-mail: pons@rect.ernet.in

When designing an assembly line, the following restrictions must be imposed on the grouping of work elements [2].

1. Precedence relationship.
2. The number of work elements cannot be greater than the number of work stations. The minimum number of workstations is one.
3. The cycle time (amount of time available at each station as well as the time between successive units coming off the line) is greater than or equal to the maximum of any station time and of the time of any work element T_i. The station time should not exceed the cycle time.

1.1 Types of Simple ALB Problems

Simple assembly line balancing problems are classified into two types, type I and type II [3]. In type I problems, the required production rate (i.e. cycle time), assembly tasks, tasks times, and precedence requirements will be given. Our objective is to minimise the number of workstations. A line with fewer stations results in lower labour costs and reduced space requirements. Type I problems generally occur when designing new assembly lines. For this purpose, to achieve the forecast demand the number of workstations should be reduced. For expansion (when demand is increased) we can also use this type I problem, to minimumise the number of extra stations to be installed.

In type II problems, when the number of workstations or production employees is fixed, the objective is to minimise the cycle time. This will maximise the production rate. Type II balancing problems generally occur, when the organisation wants to produce the optimum number of items by using a fixed number of workstations without purchasing new machines or without expansion. Here, we can identify precedence, and also zoning constraints. While balancing the main line, we have also to consider subassembly lines.

Type I problems are more common than type II. The exact algorithms available become intractable when the problem size increases. While reasonable progress has been made in the development of exact or optimal approaches, considerable advances have been reported in the development of heuristic

or inexact approaches to solve the single model deterministic task times problem. The reason is perhaps twofold: the understanding that highly efficient optimal approaches do not exist; and the need to solve large problems. Most of the inexact approaches use priority ranking or tree search logic. The model developed in this paper is to solve Type I problems. The word problem hereafter used in this text refers to Type I problems.

ALB problems fall into the NP- hard class of combinatorial optimisation problem [4]. Much work has been directed towards the development of heuristic algorithms (e.g. [3,5–9]). A comprehensive review of ABL literatures has been carried out by many researchers (e.g. [1,10,11]). The primary purpose of this paper is to propose a multi-objective genetic algorithm, and evaluate this algorithm by comparing it with available heuristic algorithms in the literature based on various performance measures.

2. Multi-Objective Genetic Algorithm

The objective of minimising the number of workstations is often employed as a criterion for assembly line balancing. Various heuristic approaches as well as optimisation techniques (integer [12], dynamic programming [13], branch and bound techniques [8,14]) have been proposed for minimising the number of workstations. These studies treated a single objective, but many real-world problems involve multiple objectives.

Genetic algorithms (GA) have been applied mainly to single objective optimisation problems. When a single objective GA is applied to a multi-objective optimisation problem, multiple objective functions should be combined into scalar fitness functions. If a constant weight is assigned to each of the objective functions to combine them, the direction of search in the GA is constant in the multi-dimensional objective space [15].

2.1 Terms and Definitions

Some of the important terms used in GA are explained in this section [16].

Chromosome: This is the complete genetic description of an individual. It is the collection of primitive features called genes.

Gene: This is a single feature within a chromosome. It may take on any of several values called alleles.

Allele: This is a particular value that may be taken on by a gene. Different genes will in general have different alleles. For example, the gene that determines hair colour may have alleles of red, black, brown, etc.

Population: Number of chromosomes forming a single population.

Objective: This is the function that is considered for minimisation or maximisation of a certain criterion.

Fitness: This is a measure of how well a parameter set performs.

The exact nature of genetic optimisation is still open to debate. Some practitioners follow binary representation within a chromosome. In this work, the chromosomes are represented by decimal numbers.

2.2 Principle Behind GA

A genetic algorithm is a set of procedures which, when repeated, enables solutions to be found for the specific problems. To achieve the desired objectives, GAs generate successive populations of alternate solutions until a solution is obtained that yields acceptable results. Within the generation of each successive population, improvements in the quality of the individual solutions are achieved. In this manner, a GA can quickly move to a successful outcome without need to examine every possible solution to the problem. The procedure used is based on the fundamental processes that control the evaluation of biological organisms, namely, natural selection and reproduction. These two processes together improve an organism's ability to survive within its environment in the following manner:

1. Natural selection determines which organisms have the opportunity of reproduction and survival within a population.
2. Reproduction involves genes from two separate individuals combining to form offspring that inherit the survival characteristics of their parents.

The GA seeks to imitate the way in which beneficial genes reproduce themselves through successive populations and hence contribute to the ability of an organism to survive.

2.3 General Scheme of GA

There are many functions for evaluating the objective criterion to measure an individual's fitness. Many methods of reproduction and mutation exist. Even the basic processes of birth and death can vary. However, for genetic optimisation the following steps are generally followed:

Initialisation: Generate the initial population randomly.

Evaluation: Compute fitness value, which is a measure of how well the individual optimises the function. Test each individual using the objective function.

Parent selection: Choose pairs of individuals from the population in such a way that those with higher fitness will get more copies.

Reproduction: Generate children from each pair of parents. Each parent contributes a portion of its genetic make-up to each child.

Mutation: Randomly changes a tiny amount of genetic information in each child.

A complete pass through the above steps is known as one generation. After each generation is complete, a new one starts with the evaluation of each of the children.

2.4 Selection Procedure

One of the simplest methods for combining multiple objective functions into a scalar fitness solution is the following weighted

Table 1. List of heuristic rules used while representing genes of chromosomes.

Rule number	Heuristic rule	Basis for determining the task priority	Reference number
1.	Maximum ranked positional weight	$RPW_i = t_i + \sum_{j \in s_i} t_j$	[5]
2.	Maximum total number of follower tasks	NS_i	[12]
3.	Maximum task time	t_i	[6]
4.	Maximum number of immediate follower tasks	NIS_i	[17]
5.	Maximum backward recursive	max (sum of the task times for i and all tasks in paths positional weight having i as its root)	[2]
6.	Minimum total number of predecessor tasks	NPS_i	[2]
7.	Minimum reverse positional weight	min (sum of the task times for i and all tasks that precede it)	[2]
8.	Minimum lower bound	$LB_i = \left[\left(t_i + \sum_j p_i\, t_j\right)/C\right]$	[12]
9.	Minimum upper bound	$UB_i = N + 1 - \left[\left(t_i + \sum_{j \in s_i} t_j\right)/C\right]^+$	[12]
10.	Minimum slack	$UB_i - LB_i$	[12]
11.	Minimum task number	Task number, i	[18]
12.	Random task assignment	Random (uniform)	[18]
13.	Maximum task time of follower task	max {Task time, S_i}	*
14.	Maximum positional weight of follower task	max {PW (S_i)}	*

*newly introduced heuristic rules
C = station cycle time
N = number of tasks to be balanced into stations
NS_i (NP_i) = total number of tasks which succeed (precede) task i (i.e. the number of elements of $S_i(P_i)$
$NIS_i(NIP_I)$ = number of tasks which must immediately succeed (precede) task i.
$S_i(P_i)$ = set of tasks which must succeed (precede) task i.
t_i = assembly time required to complete task i.
$[X]^+$ = smallest integer greater than or equal to X

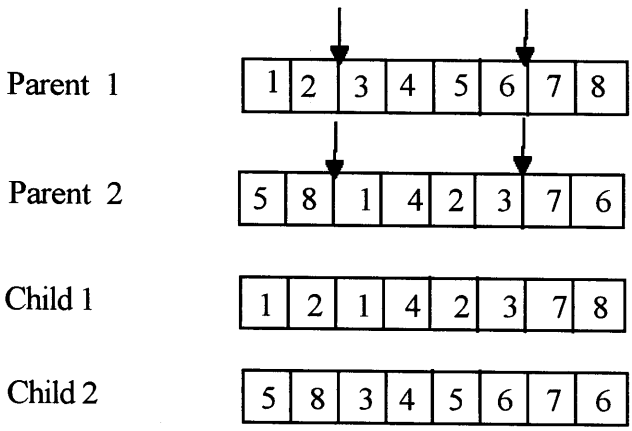


Fig. 1. Crossover operation explained.

sum approach [15]. If there are n objective functions to be maximised, the combined fitness function $F(x)$ is represented by:

$$F(x) = w_1 f_1(x) + \ldots w_i f_i(x) + \ldots + w_n f_n(x) \quad (1)$$

where, x is a string (i.e. solution)
$F(x)$ is a combined fitness function
f_i (x) is the ith objective function
w_i is a constant weight for $f_i(x)$
n is the number of objective functions

If constant weights are used to calculate $F(x)$, the search direction in the genetic algorithm is also constant. Therefore, the idea proposed by Murata et al. [15] is extended in this study for scheduling assembly lines. The w_i terms are randomly generated using the relation:

$$w_i = \frac{RN_i}{\sum_{j=1}^{n} RN_j} \quad (i = 1,2,\ldots,n) \quad (2)$$

where RN_i and RN_j are non-negative random integers. The n random real numbers generated for the weights w_i are used to calculate the weighted sum $F(x)$, when each pair of strings is selected. This procedure is iterated $N_{\text{selection}}$ (number of selections in each generation) times in each generation for selecting $N_{\text{selection}}$ pairs of parent strings for a crossover operation. The weighted sum $F(x)$ is used to determine the selection probability of each string. The w_i terms are not constant but variable, the selection probability of each string is also variable even in a single generation. This results in various search directions in the multi-objective genetic algorithm [15].

The weighting scheme in Eqs (1) and (2) means that different weights are used for selecting each pair of parent strings. Since $N_{selection}$ pairs of parent strings are to be selected in each generation, $N_{selection}$ pairs of different weights are specified using Eq. (2). If the objective functions should be minimised then use the negative sign.

2.5 Elite Preservation Strategy

In multi-objective optimisation problems, a solution with the best value of each objective can be regarded as an elite individual. Therefore, we have n elite individuals for an n-objective problem. It is natural to think that such solutions are to be preserved for the next generation in genetic algorithms.

During the execution of our multi-objective genetic algorithm, a tentative set of pareto optimal solutions are stored and updated at every generation. Here also, a certain number of individuals randomly selected from the tentative set of pareto optimal solutions are preserved, in addition to the n elite individuals, with respect to n objectives. That is, multiple Pareto optimal solutions are used as elite individuals in our multi-objective genetic algorithm.

In each and every generation, seven solutions are selected as elite individuals for a four-objective assembly line balancing problem (four elite individuals with respect to four objectives and three randomly selected individuals from the tentative set of pareto optimal solutions). These solutions are inherited by the next generation.

2.6 Multiple Objectives Considered

In this paper, the performance measures considered are, the number of stations generated, the smoothness index before the trade and transfer phase, the smoothness index after the trade and transfer phase, the line efficiency before the trade and transfer phase and the line efficiency after the trade and transfer phase.

3. The ALB Heuristic using Multi-objective GA

The step by step procedure is given below.

1. Read the data (task number, task time, cycle time, precedence tasks).
2. Compute the weights for each task using 14 heuristic rules (the heuristic rules considered are given in Table 1).
3. Rank the tasks based on the weights computed in step 2. Give the same rank for the tasks whose weights are equal.
4. Initialise the population randomly. Each gene in a chromosome represents one heuristic rule. Here the chromosome length is 14.
5. Assign the tasks to workstations using the task ranking obtained in step 3. If a tie occurs, resolve it using the heuristic rule number represented by the genes, step by step. After 14 genes are exhausted, start again from the first gene. Similarly, assign the tasks to workstations using the remaining chromosomes.
6. Calculate the values of the objective functions for the generated strings. Then, calculate the scalar fitness value $F(x)$ for each chromosome (string) by using the weights defined in Eqs (1) and (2).
7. Check for the seven best performing chromosomes, called Pareto optimal solutions, and store these for replacing the seven worst chromosomes in the next generation (five chromosomes for five optimum objective function values and two chromosomes for two best scalar fitness function values). If it is not the first iteration then replace the seven worst performing chromosomes with the Pareto optimal chromosomes stored in the previous iteration. The chromosomes are arranged in descending order based on scalar fitness function values and the last seven chromosomes are considered for replacement.
8. Select a pair of strings from the current population according to the following selection probability. The selection probability $P(x)$ of a string x in a population Ψ is specified as:

$$P(x) = \frac{f(x) - f_{\min}(\Psi)}{\sum_{x \in \Psi} \{f(x) - f_{\min}(\Psi)\}}$$

where, $f_{\min}(\Psi) = \min\{f(x)/x \in \Psi\}$

Once the selection probabilities for the chromosomes are calculated, the chromosomes are selected for the next generation using the roulette wheel approach [16].
9. Apply a crossover operation for each selected pair with a crossover probability, p_c. The two-point crossover is employed in this study [16]. This is explained by the example in Fig. 1.
10. Apply a mutation operation for each gene in the string generated by the crossover operation with a mutation probability, p_m. The insertion operator is used in this study [16]. This is explained by the example in Fig. 2.
11. Remove the seven worst (number of elite individuals considered in this study for each generation) strings from the current population and add the same number of strings from a tentative set of pareto optimal solutions.
12. If the termination condition is satisfied, stop. Otherwise go to step 5.

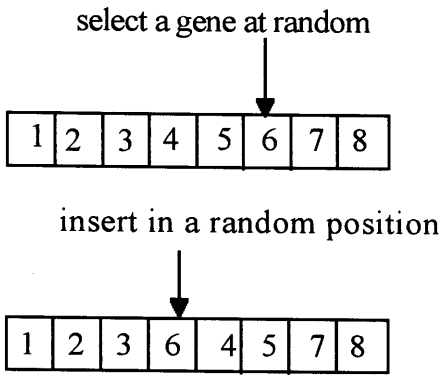
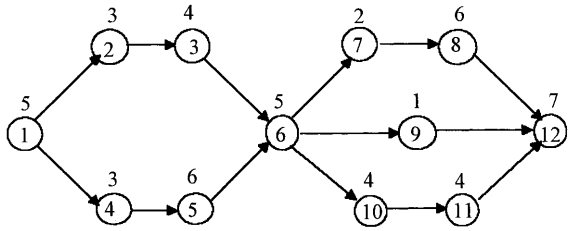


Fig. 2. Mutation operation explained.



Cycle Time = 10

Task No.	Task Time	Immediate predecessor tasks	Task No.	Task Time	Immediate predecessor tasks
1	5	------	7	2	6
2	3	1	8	6	7
3	4	2	9	1	6
4	3	1	10	4	6
5	6	4	11	4	10
6	5	3, 5	12	7	8, 9, 11

Fig. 3. Precedence diagram of assembly network for illustration.

4. Numerical Illustration

An example problem with 12 tasks and a cycle time of 10 units is considered for illustration. The problem network is shown in Fig. 2 and the step by step computations is shown below.

Step 1. Read the data for the problem shown in Fig. 3.

Step 2. Calculation of positional weights of tasks using 14 different heuristic rules, see Table 2. The positional weights are calculated using the relations in Table 1. For explanation please refer to Elsayed and Boucher [2].

Step 3. Ranking the tasks based on the positional weights. The ranking of the tasks based on heuristic rules is given in Table 3. Each row represents the ranking of 12 tasks, e.g. by using rule 1 (ranked positional weight) first task = rank 1, second task = rank 3, and so on.

Step 4. Population initialisation. Randomly generate 20 chromosomes, each having 14 genes. Each number represents the heuristic rule number. For example, 0 = ranked positional weight rule, 1 = reverse positional weight rule, and so on. The initial population is given in Table 4.

Table 3. Ranking of tasks based on positional weights.

1	3	5	2	4	6	7	8	10	7	9	11
1	2	3	2	3	4	5	6	6	5	6	7
3	5	4	5	2	3	6	2	7	4	4	1
2	3	3	3	3	1	3	3	3	3	3	4
1	3	5	2	4	6	7	8	10	7	9	11
1	2	3	2	3	4	5	6	5	5	6	7
1	2	3	2	4	5	7	9	6	8	9	10
1	1	2	1	2	3	3	4	3	3	4	5
1	2	2	2	2	3	4	4	5	4	4	5
1	2	1	2	1	1	2	1	3	2	1	1
1	2	3	4	5	6	7	8	9	10	11	12
6	9	5	2	7	1	10	8	11	4	3	6
5	4	3	2	3	4	2	1	1	4	1	6
1	3	4	2	4	5	6	8	8	7	8	9

Step 5. Assign the tasks to workstations. If a tie occurs, resolve it using the heuristic rule number represented by the genes step by step. After 14 genes are exhausted, start again from the first gene. Similarly, assign the tasks to workstations using the remaining 19 chromosomes.

Step 6. Calculation of objective function values and scalar fitness function values for each chromosome. Obtain these values (solutions) by allocating tasks to the workstations using each chromosome. Twenty chromosomes give 20 solutions. Then compute the scalar fitness function value of each solution by combining the objective function values. The objective function values and scalar fitness function values are given Table 5.

Step 7. Check for seven pareto optimal chromosomes in the population, based on the optimum objective function values and scalar fitness function values. In the subsequent generation replace the seven worst performing chromosomes with pareto optimal solutions. The pareto optimal solutions in the present generation are given below. The pareto optimal solutions are given in Table 6. The tie in selecting a chromosome is resolved arbitrarily.

Table 2. Positional weights of tasks for the 14 heuristic rules.

Number	Heuristic rule	Positional weight of tasks											
1	Max positional ranked weight	34	27	24	29	26	20	15	13	8	15	11	7
2	Max total number of follower tasks	11	8	7	8	7	6	2	1	1	2	1	0
3	Max task time	5	3	4	3	6	5	2	6	1	4	4	7
4	Max number of immediate follower tasks	2	1	1	1	1	3	1	1	1	1	1	0
5	Max backward recursive PW	132	60	57	62	59	53	15	13	8	15	11	7
6	Min total number of predecessor tasks	0	1	2	1	2	5	6	7	6	6	7	11
7	Min reverse positional weight	5	8	12	8	14	19	21	27	20	23	27	34
8	Min lower bound	1	1	2	1	2	3	3	4	3	3	4	5
9	Min upper bound	8	9	9	9	9	10	11	11	12	11	11	12
10	Min slack	7	8	7	8	7	7	8	7	9	8	7	7
11	Min task number	1	2	3	4	5	6	7	8	9	10	11	12
12	Random task assignment	43	33	50	91	40	93	30	36	26	55	57	43
13	Max task time of follower task	3	4	5	6	5	4	6	7	7	4	7	0
14	Max PW weight of follower task	29	24	20	26	20	15	13	7	7	11	7	0

Table 4. Initial population generated randomly.

Initial population of 20 chromosomes														
1	13	2	8	4	10	10	4	4	13	7	8	3	6	1
2	12	1	10	2	10	11	6	9	4	13	1	8	12	12
3	4	13	7	12	3	13	1	9	10	4	12	6	7	6
4	9	0	13	6	4	3	6	4	9	12	13	0	8	7
5	1	8	3	9	9	9	8	0	12	12	10	0	7	11
6	3	2	6	10	12	9	6	9	1	13	11	1	5	3
7	10	7	8	2	3	5	3	3	11	1	12	11	11	5
8	9	2	12	8	5	8	6	11	9	1	10	0	11	4
9	13	5	1	9	6	12	0	11	12	1	10	12	11	0
10	1	8	3	11	2	9	2	8	10	2	6	11	5	10
11	6	13	6	8	2	1	3	5	6	1	1	2	8	13
12	8	6	8	10	8	7	10	4	7	10	9	13	5	3
13	9	12	7	13	12	5	7	2	1	8	0	7	11	8
14	9	6	3	1	12	8	13	12	1	2	13	3	1	2
15	6	4	10	7	7	1	7	9	1	11	1	8	10	7
16	10	10	7	2	0	11	11	5	7	6	4	7	12	2
17	1	8	2	7	4	10	0	12	11	3	11	3	0	12
18	9	9	3	5	12	4	4	1	13	9	11	10	6	13
19	0	2	12	4	5	4	3	113	12	8	11	3	7	4
20	13	0	4	11	7	2		7	10	4	8	13	12	8

Table 5. The objective function and scalar fitness functin values.

C.No.	NOS	LEBT	SIBT	LEAT	SIAT	SFF_VALUE
1	6	83.33	5.47	92.59	2	−1.43
2	7	71.42	8.36	79.36	5.57	−2.85
3	7	71.42	8.83	89.28	2.82	−2.08
4	6	83.33	5.47	92.59	2	−1.43
5	7	71.42	8.83	89.28	2.82	−2.08
6	6	83.33	4.69	92.59	2	−1.34
7	7	71.42	8.36	79.36	5.56	−2.84
8	7	71.42	8.83	89.28	2.82	−2.08
9	6	83.33	5.47	92.59	2	−1.43
10	7	71.42	8.83	89.28	2.82	−2.08
11	6	83.33	5.47	92.59	2	−1.43
12	7	71.42	8.83	89.28	2.82	−2.08
13	7	71.42	8.36	79.36	5.56	−2.84
14	7	71.42	8.83	89.28	2.82	−2.08
15	6	83.33	4.69	92.59	2	−1.34
16	7	71.42	8.83	89.28	2.82	−2.08
17	6	83.33	5.47	92.59	2	−1.43
18	6	83.33	5.47	92.59	2	−1.43
19	6	83.33	5.47	92.59	2	−1.43
20	7	71.42	8.83	89.28	2.82	−2.08

C.No., chromosome number; NOS, number of stations created; LEBT, line efficiency before trade and transfer phase; LEAT, line efficiency after trade and transfer phase; SIBT, smoothness index before trade and transfer phase; SIAT, smoothness index after trade and transfer phase; SFF_VALUE, scalar fitness function value.

Step 8. The construction of the roulette wheel and the selection of chromosomes for intermediate population is explained in Tables 7 and 8.

Steps 9 and 10. Perform crossover and mutation. The new population after performing crossover and mutation is given in Table 9.

Step 11. Remove the worst seven strings (based on scalar fitness function values) from the current population and add the pareto optimal solutions to the current population.

Step 12. Check for termination condition. The number of generations considered is 30. The solution obtained after 30 generations is given below.

Solution before applying trade and transfer phase

The solution obtained before applying trade and transfer phase is given in Table 10.

Line efficiency (LE) = 83.33%

Smoothness index (SI) =

$\sqrt{(2^2 + 1^2 + 0^2 + 2^2 + s^2 + 3^2)} = 4.69$

Trade and transfer phase details

The transfer phase details [7] are:

GOAL = 1.5 LTS = 3 STS = 6 TASK = 9

where

$\text{GOAL} = \frac{ST_{\max} - ST_{\min}}{2}$

$ST_{\max}$ = maximum station time

$ST_{\min}$ = minimum station time

LTS = largest time station

STS = smallest time station

TASK = task in the LTS with least task time

This means we are transferring task 9, from station 3 to station 6. There are no other possibilities for transfer.

The trade phase details are:

GOAL = 0.5 LTS = 2 STS = 1 LST = 2 SST = 4

where,

LST = largest time station task with minimum task time

SST = smallest time station task with minimum task time

This means we are interchanging task 2 (from largest time station 2) with task 4 in the smallest station time station 1. Here, we are not violating the precedence restrictions. No other possibilities exist for trading.

Trade and transfer between ranked stations

In this case, this possibility is not present. In the above assignment the maximum station time is 9.0, So, set the cycle time to 9.0, without violating any constraints. The solution after applying the trade and transfer phase is given in Table 11. The line efficiency and smoothness index are given below.

Line efficiency = 92.59%

Smoothness index = $\sqrt{(1^2 + 0^2 + 0^2 + 1^2 + 1^2 + 1^2)} = 2.00$

5. Comparison of the algorithms

In order to generalise the results, both problems taken from open literature and randomly generated problems are solved. Six popular heuristic algorithms and the genetic algorithm are tested for 100 problems (20 networks and each solved for 5 different cycle times). The number of tasks varies from 7 to 50. Details of this data set are shown in Table 12.

Table 6. Pareto optimal solutions.

Chromosome selected														Objective function
13	2	8	4	10	10	4	4	13	7	8	3	6	1	NOS
13	2	8	4	10	10	4	4	13	7	8	3	6	1	LEBT
3	2	6	10	12	9	6	9	1	13	11	10	5	3	SIBT
13	2	8	4	10	10	4	4	13	7	8	3	6	1	LEAT
13	2	8	4	10	10	4	4	13	7	8	3	6	1	SIAT
8	6	8	10	8	7	10	4	7	10	9	13	5	3	Random
13	0	4	11	7	2	1	7	10	4	8	13	12	8	Random

Table 7. Roulette wheel construction and chromosomes selected.

C.No.	$P(x)$	$CP(x)$	R.No.	Ch. Sel.
1	0.08	0.08	0.95	19
2	0	0.08	0.8	17
3	0.04	0.12	0.59	14
4	0.08	0.19	0.34	8
5	0.04	0.23	0.73	16
6	0.08	0.31	0.89	19
7	0	0.31	0.06	1
8	0.04	0.35	0.93	19
9	0.08	0.42	0.44	10
10	0.04	0.46	0.25	6
11	0.08	0.54	0.14	4
12	0.04	0.58	0.1	3
13	0	0.58	0.77	17
14	0.04	0.62	0.73	16
15	0.08	0.7	0.08	3
16	0.04	0.74	0.07	1
17	0.08	0.81	0.63	15
18	0.08	0.89	0.43	10
19	0.08	0.96	0.59	14
20	0.04	1	0.72	16

C.no., chromosome number; $P(x)$, probability of selection; CP(x), Cumulative probability; R.no., random number, Ch. Sel. chromosome selected.

Table 8. The intermediate population.

Intermediate population of 20 chromosomes														
1	0	2	12	4	5	4	3	13	12	8	11	3	7	4
2	1	8	2	7	4	10	0	12	11	3	11	3	0	12
3	9	6	3	1	12	8	13	12	1	2	13	3	1	2
4	9	2	12	8	5	8	6	11	9	1	10	0	11	4
5	10	10	7	2	0	11	11	5	7	6	4	7	12	2
6	0	2	12	4	5	4	3	13	12	8	11	3	7	4
7	13	2	8	4	10	10	4	4	13	7	8	3	6	1
8	0	2	12	4	5	4	3	13	12	8	11	3	7	4
9	1	8	3	11	2	9	2	8	10	2	6	11	5	10
10	3	2	6	10	12	9	6	9	1	13	11	10	5	3
11	9	0	13	6	4	3	6	4	9	12	13	0	8	7
12	4	13	7	12	3	13	1	9	10	4	12	6	7	6
13	1	8	2	7	4	10	0	12	11	3	11	3	0	12
14	10	10	7	2	0	11	11	5	7	6	4	7	12	2
15	4	13	7	12	3	13	1	9	10	4	12	6	7	6
16	13	2	8	4	10	10	4	4	13	7	8	3	6	1
17	6	4	10	7	7	1	7	9	1	11	1	8	10	7
18	1	8	3	11	2	9	2	8	10	2	6	11	5	10
19	9	6	3	1	12	8	13	12	1	2	13	3	1	2
20	10	10	7	2	0	11	11	5	7	6	4	7	12	2

Table 9. Population after crossover and mutation.

1	3	2	3	8	11	5	10	12	13	1	3	4	0	6
2	5	9	2	4	4	2	12	1	13	8	6	10	13	3
3	12	6	5	6	8	1	6	9	4	13	5	12	7	8
4	8	9	0	11	9	6	12	1	12	1	13	11	12	2
5	7	12	11	2	4	8	13	3	3	1	12	6	0	9
6	12	11	3	2	9	4	4	12	4	3	6	8	10	13
7	7	8	0	6	13	10	3	11	11	2	0	5	5	3
8	3	2	6	10	10	12	12	8	9	13	6	1	10	5
9	7	6	3	4	1	13	12	4	10	7	11	8	5	1
10	3	0	4	1	13	3	11	12	7	13	6	8	11	10
11	3	6	7	2	9	5	11	1	8	11	10	12	2	13
12	2	6	12	6	10	12	9	6	13	9	11	10	5	13
13	9	5	12	4	4	8	10	10	6	13	11	2	7	1
14	4	5	12	3	1	7	13	9	11	1	4	9	10	11
15	12	5	6	0	13	3	7	9	13	8	11	4	8	10
16	2	4	11	10	6	8	4	12	13	3	2	9	11	13
17	0	4	7	8	12	0	13	11	7	6	11	7	1	2
18	11	6	2	0	1	12	12	13	9	11	8	1	12	4
19	2	13	12	3	11	9	5	8	10	9	2	13	11	5
20	8	0	3	8	4	7	4	13	9	10	12	1	11	11

Table 10. Initial assignment of tasks before trade and transfer phase.

Station	Task number	Task time (T_e)	Station time (ST_i)	$CT–ST_i$
1	1	5	8	2
	4	3		
2	2	3	9	1
	5	6		
3				
	3	4	10	0
	6	5		
	9	1		
4	7	2	8	2
	8	6		
5	10	4	8	2
	11	4		
6	12	7	7	3

5.1 Algorithms Compared

The six popular heuristic algorithms considered for comparison are:

Ranked positional weight heuristic [5]
Moodie and Young two phase method [7]
Kilbridge and Wester heuristic [6]
Hoffmann precedence matrix procedure [9]
Immediate update first fit(IUFF) heuristic [3]
Rank and assign (RA) heuristic [3]

Table 11. Revised assignment of tasks after trade and transfer phase.

Station	Task number	Task time (T_e)	Station time (ST_i)	$CT–ST_i$
1	1	5	8	1
	2	3		
			9	0
2	4	3		
	5	6		0
			9	
3	3	4		
	6	5		
4	7	2	8	1
	8	6		
5	10	4	8	1
	11	4		
6	9	1	8	1
	12	7		

In the case of IUFF and RA algorithms, the numerical score functions considered are given in Table 13.

5.2 Performance Measures

Perfect balance of the line means the combination of the elements of the work to be done in such a manner that at each station the sum of the elemental times just equals the cycle time. When a perfect balance cannot be achieved, we measure the effectiveness of the balance by the following.

Number of Excess Stations: The number of excess stations is the measure considered by many researchers. A line with fewer workstations translates into lower labour costs and reduces the space requirements, so it will be a more cost effective plan.

Table 13. Numerical score functions considered in IUFF and RA heuristic methods.

n	Name	Description
1	Reverse positional weight	Sum of the task times for *x* and all tasks that precede it
2	Number of followers	Number of tasks that follow task *x*
3	Number of immediate followers	Number of tasks that immediately follow taxk *x*
4	Number of predecessors	Number of tasks that precede task *x*
5	Work element time	Task time of *x*
6	Backward recursive positional weight	Sum of the task times for *x* and all tasks in paths having *x* as its root

Table 12. Dataset used for comparisons.

Problem source	Number of Tasks	Cycle times					Optimal number of stations for each cycle time				
Elsayed and Boucher [2] (p. 354)	12	8	10	13	17	25	7	5	4	3	2
Hoffmann [9]	9	7	8	9	10	13	6	5	5	4	3
Hoffmann [9]	19	8	10	12	15	19	10	8	7	6	4
Elsayed and Boucher [2] (p. 348)	9	10	12	13	16	24	5	4	4	3	2
Groover [19] (p. 147)	12	0.8	0.9	1.0	1.1	1.4	5	5	4	4	3
Groover [19] (p. 169)	24	2.0	2.5	3.0	4.0	6.0	5	4	3	3	2
Groover [19] (p. 168)	14	0.65	0.8	1.0	1.1	1.3	6	5	4	4	3
Groover [19] (p. 168)	10	1.1	1.2	1.4	1.5	2.0	6	5	5	4	3
Groover [19] (p. 166)	10	0.9	1.0	1.1	1.3	1.4	5	5	4	4	3
Groover [19] (p. 167)	8	1.6	1.8	2.0	2.5	3.0	4	4	3	3	2
Askin and Standridge [20] (p. 63)	11	31	40	50	75	80	5	4	3	3	2
Gutjahr and Nemhauser [4]	9	7	8	10	12	14	6	5	4	4	3
Hoffmann [9]	7	10	11	12	14	21	5	4	4	3	2
Askin and Standridge [20] (p. 64)	9	36	40	50	75	80	5	4	3	2	2
Wee and Magazine [8]	11	0.35	0.4	0.5	0.7	1.0	6	5	4	3	2
Askin and Standridge [20] (p. 40)	12	51	60	70	80	105	4	4	3	3	2
Groover [19]	11	48	52	62	64	97	4	4	3	3	2
Brian and Patterson [12]	9	19	25	30	38	40	4	3	3	2	2
Kilbridge and Wester [6]	45	69	92	138	184	276	8	6	4	3	2
Generated problem	50	100	150	200	250	300	24	16	12	10	8

Further, if the number of stations is reduced, then capacity utilisation is generally increased.

Line Efficiency (LE): The line efficiency is the ratio between total station time to the product of cycle time and the number of workstations, represented as a percentage. It shows the percentage use of the line and is expressed as:

$$LE = \frac{\sum_{i=1}^{K} ST_i}{K \times CT} \times 100$$

where,

K = total number of workstations
CT = cycle time

Smoothness Index (SI): The smoothness index is an index for the relative smoothness of a given assembly line. A smoothness index of 0 indicates a perfect balance. A smaller SI results in a smoother line, thereby reducing the in-process inventory.

$$SI = \sqrt{\left(\sum_{i=1}^{K} (ST_{\max} - ST_i\right)}$$

Where,

$ST_{\max}$ = maximum station time
ST_i = station time of station i.

Execution (CPU) Time: Execution time is also considered by many researchers since it is directly tied to the efficiency of the algorithm selected.

5.3 Experimental Conditions and Implementation

The programs are coded in the C++ language. The computer system used is an HCL-HP Pentium with 32 MB RAM, 2 GB HDD, 133 MHz speed. All the methods are programmed to read from the same data set in an identical manner. Various object oriented programming (OOP) concepts, constructors, call by reference, pointers, dynamic memory allocation (new and delete) functions have been used in order to increase the speed of execution of the programs, and effective use of the memory. The main advantage of using class is that various information and associated functions about an entity (task or station) can stored under a name which makes addressing easy. Classes make use of memory in a dynamic fashion and can free the memory allocated during the execution of the program, making it possible for some entity to share the memory space. In all the methods we used, two classes namely "task" and "station" are used. The task class can hold all the details relating to the tasks (e.g. task numbers, task times, precedence tasks, available tasks for the assignment, and ranking of the tasks based on weights) and the associated functions. Similarly, "class station" holds details such as station number, station time, station tasks and the functions relating to assignment of the task.

Pointers have been used extensively in the program, so transfer of data between functions has been reduced to a minimum, and hence duplicate copies of the same value are avoided as far as possible. The use of pointers also increases the computational efficiency, as the data is accessed directly and not through an intermediate step.

6. Comparative Results

To evaluate the performance of the proposed multi-objective genetic algorithm compared to that of the six heuristic procedures, 20 assembly networks are each solved for five different cycle times. For the problems considered in this study, please refer to Table 12. A summary of the results is given in Table 14 and the discussions are given in the following sections.

6.1 Comparison for Various Performance Measures

Number of excess stations

Among the six heuristic algorithms, the precedence matrix procedure of Hoffmann performs best. Among the six heuristic rules considered, in the case of IUFF, the maximum task time, and the maximum recursive positional weight rules perform reasonably well. The RA method gives very poor results compared to the others. GA often gives good results when there are generations. When there are 10 generations, the results are very close to those of the Hoffmann precedence matrix procedure. Please see Fig. 4 and Table 14.

Average Line Efficiency (LE): Before Trade and Transfer Phase

In this case also, GA gives the best results. The average line efficiency obtained by GA before trade and transfer is 85.01%. Next to GA, the Hoffmann precedence matrix procedure gives good results. In case of IUFF, the maximum task time rule gives good results, which are very close to those of the ranked positional weight method. The RA heuristic performs poorly. Among the RA heuristic rules, the minimum reverse positional weight rule performs best. Please refer to Fig. 5 and Table 14.

Average Smoothness Index (SI): Before Trade and Transfer Phase

For the problems tested, GA gives a smaller smoothness index, 10.36 (this results in smoother lines, thereby reducing the in-process inventory). Next to this, the Hoffmann precedence matrix procedure gives good results with an average smoothness index of 11.54. In the case of IUFF, the maximum task time rule performs better. RA performs poorly. When the number of iterations in the GA reduces to five, the average SI is nearly the same as for the precedence matrix method. Please refer to Fig. 6 and Table 7.

Average Line Efficiency (LE): After Trade and Transfer Phase

The average line efficiency of GA, after the trade and transfer phase is 91.884%. Next to the GA, the line efficiency of the Hoffmann enumeration procedure is 90.92%. In IUFF, the maximum backward recursive positional weight performs best

Table 14. Comparative results of various heuristic algorithms.

Method	Number of excess stations	Average LEBT	Average SIBT	Average LEAT	Average SIAT	Total execution time (s)
Ranked positional weight method	59	82.66	18.84	90.36	8.22	2.25
Kilbridge and Wester heuristic	64	82.18	15.84	89.89	9.21	1.54
Moodie and Young method	68	81.29	21.93	89.59	9.37	1.54
Hoffmann precedence matrix	51	84.39	11.54	90.92	6.58	18.35
IUFF Heuristic						
1. Minimum reverse positional weight	69	81.27	19.96	89.553	9.884	1.593
2. Maximum number of follower tasks	78	79.57	23.16	88.81	10.12	1.703
3. Maximum number of immediate follower tasks	72	80.80	19.25	88.954	9.69	1.978
4. Minimum number of predecessor tasks	87	78.34	21.32	88.35	10.11	1.484
5. Maximum task time	64	82.18	15.83	89.89	9.21	1.429
6. Maximum backward recursive PW	69	81.44	22.29	89.98	9.77	1.758
R and A Heuristic						
1. Minimum reverse positional weight	101	76.93	22.11	87.87	10.61	1.590
2. Maximum number of follower tasks	106	76.36	23.63	86.92	11.76	1.39
3. Maximum number of immediate follower tasks	115	75.06	25.10	86.76	12.03	1.429
4. Minimum number of predecessor tasks	107	76.07	24.12	88.04	10.84	1.494
5. Maximum task time	106	76.25	22.50	87.04	11.73	1.246
6. Maximum backward recursive PW	105	76.68	21.91	86.64	11.47	1.538
Multi-objective genetic algorithm (GA)	50	85.01	10.36	91.88	6.21	750

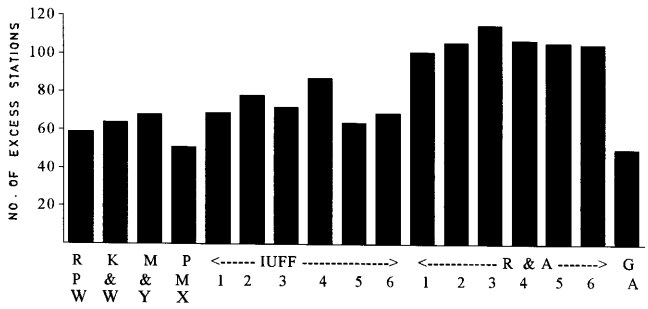


Fig. 4. Heuristic rule vs number of excess stations. RPW, Ranked positional weight method; K&W, Kilbridge – Webster method; M& Y, Moodie & Young method; PMX, Hoffmann's precedence matrix method; IUFF, Immediate update first fit method: 1 = minimum reverse positional weight rule, 2 = maximum number of follower tasks rule, 3 = maximum number of immediate follower tasks rule, 4 = minimum total number of predecessor tasks rule, 5 = maximum task time rule, 6 = maximum backward recursive positional weight rule; R&A, Rank & assign method: 1 = minimum reverse positional weight rule; 2 = maximum number of follower tasks rule; 3 = maximum number of immediate follower tasks rule; 4 = minimum total number of predecessor tasks rule; 5 = maximum task time rule; 6 = maximum backward recursive positional weight rule; GA, Genetic algorithm.

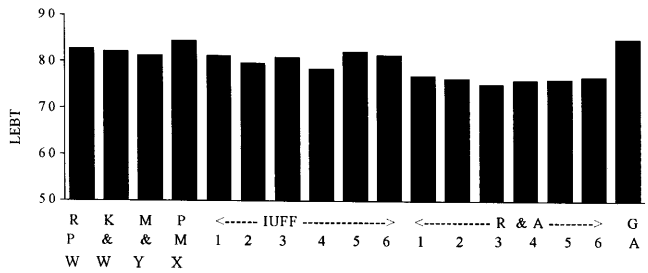


Fig. 5. Line efficiency before trade & transfer (LEBT) vs. heuristic rule (see Fig. 4 for key).

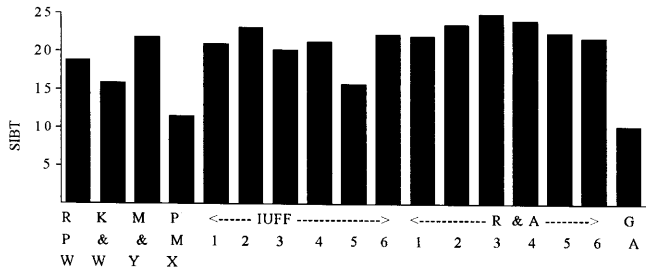


Fig. 6. Smoothness index before trade 7 transfer (SIBT) vs. heuristic rule (see Fig. 4 for key).

with an efficiency of 89.982%. Efficiencies of RA rules are also comparable with the other methods. Among the RA rules, the maximum number of predecessor tasks rule performs best with an average efficiency of 88.04%. Please refer to Fig. 7 and Table 14.

Average Smoothness Index (SI): After Trade and Transfer Phase

The average SI given by GA is 6.212 . Next to GA, the Hoffmann enumeration procedure stood second with an average

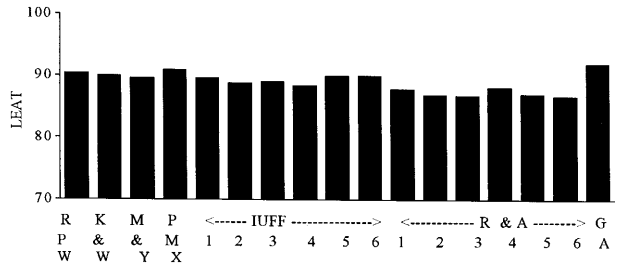


Fig. 7. Line efficiency after trade & transfer (LEAT) vs. heuristic rule (see Fig. 4 for key).

SI of 6.582. The ranked positional weight, Moodie and Young method, and Kilbridge and Wester heuristic also performs better. Please refer to Fig. 8 and Table 14.

Total Execution Time (in s)

All the RA rules are executed using minimum CPU time. Each rule takes approximately 1.50 s. For the RA with the maximum task time rule the CPU time is 1.246 s. Except for the Hoffmann precedence matrix procedure and GA, for all other rules the CPU time is less than 2 s. GA takes more time because of the increased number of generations. The execution time depends on the number of generations and the population size.

6.2 Summary

Except for the execution time criterion the proposed GA gives better results than the other heuristics considered. Among the six heuristics considered, the Hoffmann enumeration procedure gives the best results. If time taken is not important, then GA definitely gives good results.

7. Conclusions

A multi-objective genetic algorithm is proposed for solving ALB problems using genetic algorithms (GA). Fourteen heuristic decision rules are used for representing the genes of chromosomes. In this paper, two new heuristic rules are proposed. To compare the performance of GA with heuristics, the six popular heuristics available in the literature are considered. The multi-objective GA and the six heuristics are tested on 100 problems. In order to increase the line efficiency and smoothness of the line, the trade and transfer phase of the Moodie and Young method is applied to all the six heuristics and also to the proposed genetic algorithm. It is observed that the smoothness of the line depends upon the heuristic method used.

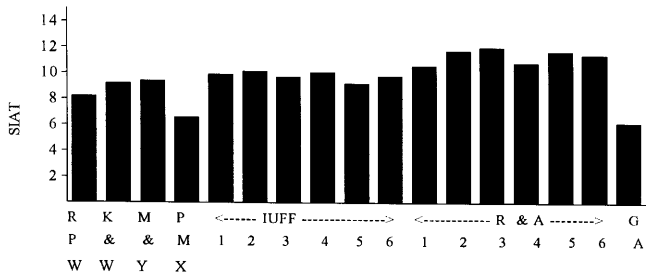


Fig. 8. Smoothness Index after trade & transfer (SIAT) vs. heuristic rule (see Fig. 4 for key).

All the methods are compared for performance measures, the number of excess workstations, the line efficiency, the smoothness index before and after the trade and transfer phase, and the CPU time. Comparative results are presented in the form of tables and charts. The proposed multi-objective genetic algorithm performs better than the other methods. Among the six considered heuristics, the Hoffmann enumeration procedure performs best. The execution time for GA is much greater, because GA searches for global optimal solutions using more iterations. Further, with GA, we can obtain multiple solutions for different objective criteria so that the user can select one best solutions from among the optimal solutions.

Acknowledgement

This work was partially supported by a research grant from the University Grants Commision, New Delhi, India, no. F.14–34/96 (SR-I), dated 5 April 1996.

References

1. Soumen Ghosh and Roger J. Gagnom, “A comprehensive literature review and analysis of the design, balancing and scheduling of assembly lines”, International Journal of Production Research, 27(4), pp. 637–670, 1989.
2. E. A. Elsayed, and T. O. Boucher, “Analasis and control of production systems”, Prentice Hall International Series in Industrial and Systems Engineering, New Jersey, 1994.
3. S. T. Hackman, M. J. Magazine, and T. S. Wee, “Fast, effective algorithms for simple assembly line balancing problems”, Journal of Operational Research, 37(6), pp. 916–924, 1989.
4. A. L. Gutjahr and G. L. Nemhauser, “An algorithm for the line balancing problem”, Management Science, 11(2), pp. 308–315, 1964.
5. W. P. Helgeson and D. P. Birnie, “Assembly line balancing using the ranked positional weight technique”, Journal fo Industrial Engineering, 12(6), pp. 394–398, 1961.
6. M. D. Kilbridge and L. Wester, “A heuristic method of assembly line balancing”, Journal of Industrial Engineering, 12(4), pp. 292–298, 1961.
7. C. L. Moodie and H. H. Young, “A heuristic method of assembly line balancing for assumptions of constant or variable work element times”, Journal of Industrial Engineering, 16(1), pp. 23–29, 1965.
8. T. S. Wee and M. J. Magazine, “An efficient Branch and Bound algorithm for an assembly line balancing problem. Part 1. Minimise the number of workstations”, Working Paper 150, University of Waterloo, Ontario, Canada, 1981.
9. T. R. Hoffmann, “Assembly line balancing with a precedence matrix”, Management Science, 9(4), pp. 551–562, 1963.
10. I. Baybars, “A survey of exact algorithms for the simple assembly line balancing problem”, Management Science, 32(8), pp. 909–932, 1986.
11. T. F. Brian, J. H. Patterson and W. V. Gehrlein, “A comparative evaluation of heuristic line balancing techniques”, Management Science, 32(4), pp. 430–454, 1986.
12. T. F. Brian and J. H. Patterson, “An integer programming algorithm with network cuts for solving the assembly line balancing problem”, Management Science, 30(1), pp. 85–99, 1984.

13. L. Scharge and K. R. Baker, “Dynamic programming solution of sequencing problems with precedence constraints”, Operational Research, 26, pp. 444–449, May–June 1978.
14. R. V. Johnson, “A branch and bound algorithm for assembly line balancing problems with formulation irregularities”, Management Science, 29(11), pp. 1309–1324, 1983.
15. T. Murata, H. Ishibuchi and H. Tanaka, “Multi-objective genetic algorithm and its application to flowshop scheduling”, Computers and Industrial Engineering, 30(4), pp. 957–968, 1996.
16. Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer-Verlag, Berlin, 1992.
17. F. M. Tonge, “Summary of a heuristic line balancing procedure”, Management Science, 7(1), pp. 21–42, 1969.
18. A. L. Arcus, “An analysis of a computer method of sequencing assembly line operations”, PhD dissertation, University of California, Berkley, 1963.
19. M. P. Groover, Automation, Production Systems, and Computer Integrated Manufacturing, Printed-Hall of India, New Delhi, 1996.
20. R. G. Askin and C. R. Standridge, Modelling and Analysis of Manufacturing Systems, John Wiley, 1993.