American Journal of Industrial and Business Management, 2016, 6, 674-696
Published Online May 2016 in SciRes. http://www.scirp.org/journal/ajibm
http://dx.doi.org/10.4236/ajibm.2016.65063

Design and Comparison of Genetic Algorithms for Mixed-Model Assembly Line Balancing Problem with Original Task Times of Models

Panneerselvam Sivasankaran[1], Peer Mohamed Shahabudeen[2]

[1]Department of Mechanical Engineering, Manakula Vinayagar Institute of Technology, Pondicherry, India
[2]Department of Industrial Engineering, College of Engineering, Anna University, Chennai, India
Email: sivasankaran.panneerselvam@yahoo.com, psdeen@gmail.com

Received 30 April 2016; accepted 28 May 2016; published 31 May 2016

Abstract

Assembly line balancing is a key for organizational productivity in terms of reduced number of workstations for a given production volume per shift. Mixed-model assembly line balancing is a reality in many organizations. The mixed-model assembly line balancing problem comes under combinatorial category. So, in this paper, an attempt has been made to develop three genetic algorithms for the mixed-model assembly line balancing problem such that the combined balancing efficiency is maximized, where the combined balancing efficiency is the average of the balancing efficiencies of the individual models. At the end, these three algorithms and another algorithm in literature are compared in terms of balancing efficiency using a randomly generated set of problems through a complete factorial experiment, in which "Algorithm", "Problem Size" and "Cycle Time" are used as factors with two replications under each of the experimental combinations to draw inferences and to identify the best of the four algorithms. Then, through another set of randomly generated small and medium size data, the results of the best algorithm are compared with the optimal results obtained using a mathematical model. It is found that best algorithm gives the optimal solution for all the problems in the second set of data, except for one problem which cannot be solved using the model. This observation supports the fact that the best algorithm identified in this paper gives superior results.

Keywords

Assembly Line Balancing, Cycle Time, Genetic Algorithm, Crossover Operation, Mixed-Model, Mathematical Model

How to cite this paper: Sivasankaran, P. and Shahabudeen, P.M. (2016) Design and Comparison of Genetic Algorithms for Mixed-Model Assembly Line Balancing Problem with Original Task Times of Models. *American Journal of Industrial and Business Management*, **6**, 674-696. http://dx.doi.org/10.4236/ajibm.2016.65063

1. Introduction

The mass production system aims to maximize the production volume per shift with given resources or minimize the number of workstations for a given production volume per shift. In industries, product line (line layout) is used for the mass production system. If a shop manufactures individual component using a line layout, then the machines that are required are arranged as per the process sequence of that component. If a shop assembles components and subassemblies to have an end product, the tasks that are performed to assemble the end product possess precedence relationships amongst them. The objective of this system of producing assemblies is to balance the assembly line such that the balancing efficiency is maximized.

The assembly line balancing problem is basically classified into *ALBP* 1 and *ALBP* 2. The *ALBP* 1 is the type 1 assembly line balancing problem in which the objective is to group the tasks into a minimum number of workstations for a given cycle time, which in turn maximizes the balancing efficiency of the assembly line. The *ALBP* 2 is the type 2 assembly line balancing problem, in which the tasks are grouped into a given number of workstations such that the cycle time (the maximum of the sum of the task times of the workstations) is minimized. This in turn maximizes the production volume per shift.

If a company has more than one model of a product, then setting up a separate assembly line will lead to more workstations and in turn more assemblers with less utilized workstations. So, if more models of the product are assembled in the same assembly line, the total number of workstations to assemble all the models of the product will be reduced and at the same time each workstation will be assigned with the tasks of models to utilize the assigned cycle time to that workstation maximally.

In the single model assembly line balancing problem, there will be one cycle time, where as in the mixed-model assembly line balancing problem, each model will have a cycle time which is normally computed based on the desired production volume per shift of that model. In a problem with two models, if the production volume of Model 1 is 20 assemblies per shift, then the corresponding cycle time is 24 minutes and if that of Model 2 is 40 assemblies per shift, then the corresponding cycle time is 12 minutes. In the case of single model, the cycle time will be considered as such without any modification for each of the models. In the mixed-model assembly line balancing problem, the line will be designed for a common cycle time which is computed based on the cycle times of the individual models. If the cycle times of the models are one and the same, then the common cycle time will be equal to each of them; otherwise, the average of the cycle times of the models may be treated as the common cycle time for the mixed-model assembly line balancing problem.

Consider Model 1 and Model 2, whose precedence networks are shown in **Figure 1** and **Figure 2**, respectively. The number by the side of each node represents the respective task time. **Figure 3** shows the precedence network of the combined model, which consists of the tasks of both models with combined precedence relationships. The set of tasks in the combined model is the union of the tasks of the individual models. While assigning each task to each workstation of the mixed model assembly line, the precedence network of the combined model shown in the **Figure 3** is used with concurrent reference to the precedence networks of the individual models to know the presence of that activity in one of the models or both the models as well as its task times in those models.

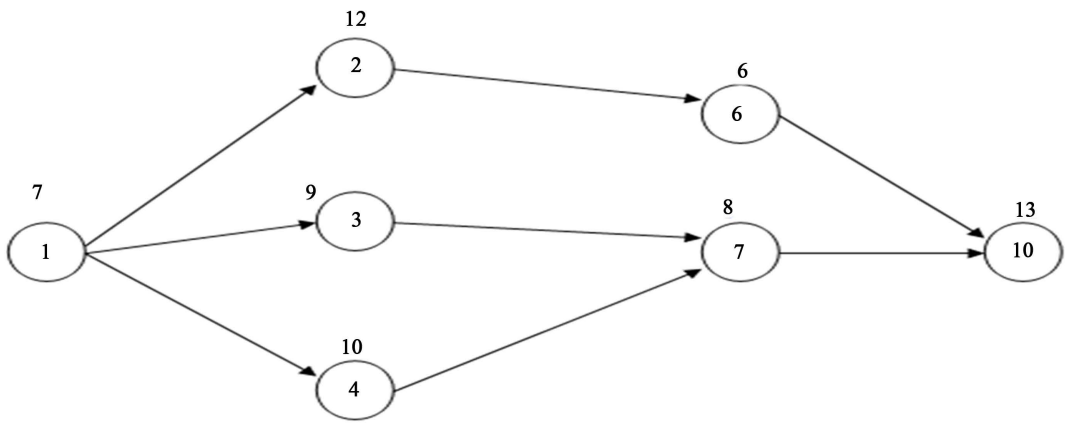

Figure 1. Precedence network of Model 1.

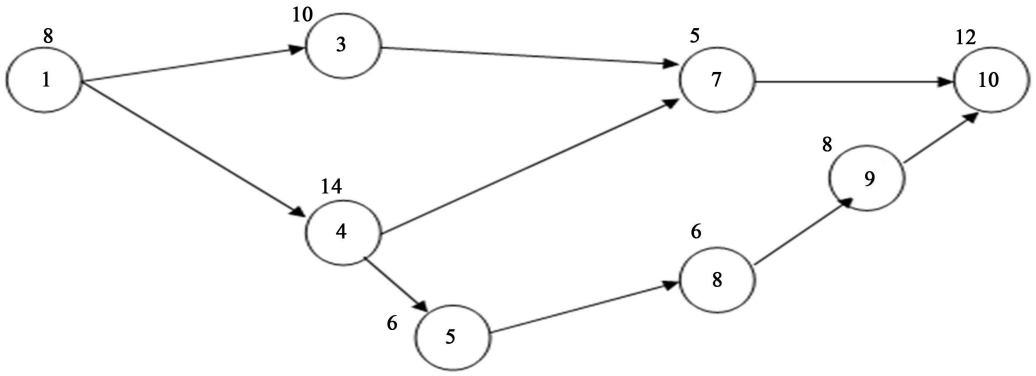

Figure 2. Precedence network of Model 2.

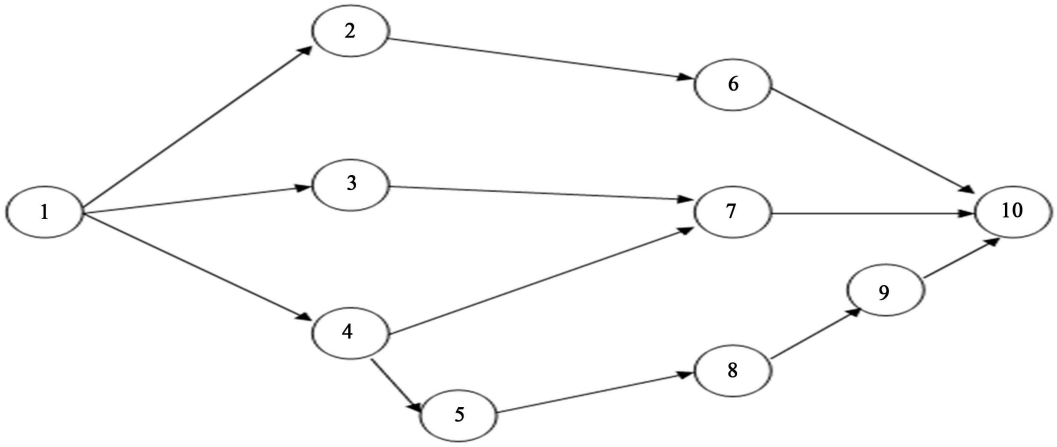

Figure 3. Combined model of Model 1 and Model 2.

Researchers in the past except Sivasankaran and Shahabudeen [1] computed average task time for each of the tasks in the combined model. But, in this work, the average task time is not computed. Instead, the original task times of the models are used as such in the design of the assembly line without any modification, because it introduces perfection in the design of the assembly line in terms of reduced number of workstations.

2. Literature Review

Assembly line balancing problems are classified into eight types, which based on the following attributes.

- Number of models (Single model/Mixed-model)
- Nature of task times (Deterministic/Probabilistic)
- Nature of flow (Straight type/U type)

If only a single model is assembled in an assembly line, then the production system is defined as a single model assembly system; otherwise, it is called a mixed-model assembly system. If the task times are assumed to constant, then that situation is called as deterministic task time situation; otherwise, it is called as probabilistic task time situation, in which the task times are represented in the form of a fitting probability distribution. The arrangement of the workstations of the assembly line may be in a straight line layout or in a U shape layout [1] [2]. In the U shape layout, an operator may manage more than one workstation. The corresponding classification of the assembly line balancing problems is shown in **Figure 4** (Sivasankaran and Shahabudeen [2]).

This section presents the review of literature of the mixed-model straight type assembly line balancing problem. Readers may refer Sivasankaran and Shahabudden [2] for details of review of other types of assembly line balancing problems.

In this paper, the contributions of the researchers in the area of mixed-model assembly line balancing problems are classified into the following categories.

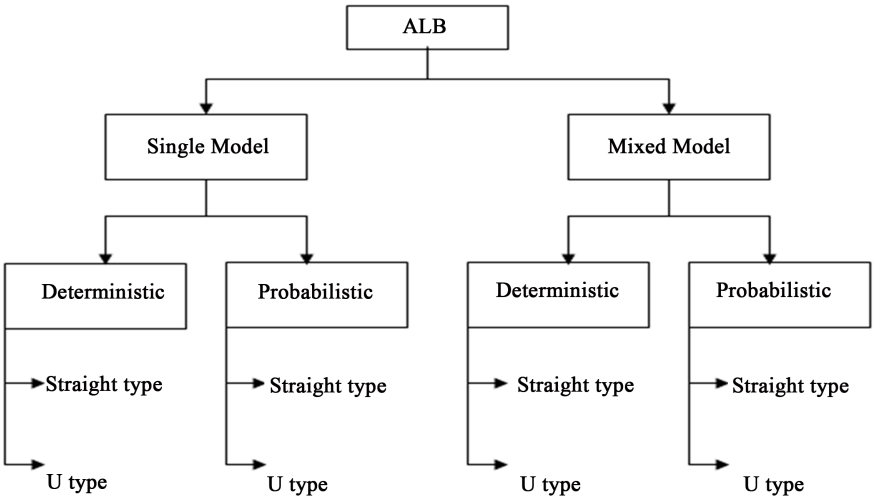

Figure 4. Classification of ALB problems (Sivasankaran and Shahabudeen [2]).

- Mathematical models
- Branch and bound method
- Heuristics
- Genetic algorithms
- Simulated annealing algorithms
- ACO algorithm

2.1. Mathematical Models

Gokcen and Erel [3] developed a goal programming model to the mixed-model assembly line balancing problem. In this model, they introduced assignment constraints, precedence constraints, cycle time constraint, zoning constraints and station constraints. It is illustrated with an example problem. Gokcen and Erel [4] developed a binary integer formulation for the mixed-model assembly line balancing problem in which the number of stations is minimized for given cycle times of the models. Choi [5] developed a goal programming model for the mixed-model assembly line balancing problem for balancing processing time and physical workload at the same time. Emde *et al.* [6] carried out a computational evaluation of objectives to smoothen workload in mixed-model assembly line balancing problem. The results of the study reveal that workload smoothing is an essential task in the mixed-model assembly lines. Kara *et al.* [7] developed a multi-objective mathematical model to balance the mixed-model assembly line for a model mix having precedence conflict and duplicate common tasks. Sivasankaran and Shahabudeen [8] developed a hybrid mathematical model for single model assembly line balancing problem. In the first phase, they presented a mathematical model to design the workstations such that the balancing efficiency is maximized for a given cycle time and in the second phase, another mathematical model is presented to minimize the cycle time for the number of workstations which is obtained in the first phase.

Ozturk *et al.* [9] considered the flexible mixed-model assembly line balancing problems with parallel workstations with the objective of designing the assembly lines to minimize the number of workstations under demand fluctuations. They developed a mathematical model for this problem and a decomposition scheme for this problem. Sekar [10] developed multi-objective mixed-integer programming (MOMIP) model to minimizing the work overload and station-to-station product flows in the mixed-model assembly line balancing problem. Akpinar and Baykasoglu [11] [12] considered the mixed-model assembly line balancing problem with parallel workstations, zoning constraints and sequence dependent setup times between tasks. They developed a mathematical model and solved it using a hybrid meta-heuristic. Part II of their contributions deals with the development of a multiple colony hybrid bees algorithm.

Though mathematical models give the best solution (optimal solution) for the assembly line balancing problem, its application in practice is very much limited, because the number of variables and the number of constraints that can be used in the software that solves the models are limited in the form of upper limit. So, there use is limited to very small size problems.

2.2. Branch and Bound Method

Bukchin and Rabinowitch [13] developed a branch and bound based solution approach for the mixed-model assembly line balancing problem to minimize the number of workstations and task duplication costs. Emde *et al.* [6] considered the mixed model assembly line balancing problem with the objective of maximizing balancing efficiency along with work load smoothness. They proposed 28 workload smoothing criteria and investigated them using branch and bound algorithm and found that they play a vital role in planning and executing the mixed-model assembly line balancing problem. Yang *et al.* [14] considered the mixed-model assembly line balancing problem in which the variants of a task to different models should be assigned to an identical workstation is relaxed. That is variants of a task over different models can be duplicated on two adjacent workstations to improve the balancing efficiency. They developed a branch, bound and remember algorithm to solve the problem.

When compared to mathematical models for the assembly line balancing problems, the branch and bound algorithm can solve relatively large size problems. But, if the data is such that all the nodes in the branching tree are to be explored before finding the optimal solution, then the time taken to solve an assembly line balancing problem using this method may coincide with that using the corresponding mathematical model.

2.3. Heuristics

Kim and Kim [15] considered the mixed-model assembly line balancing problem. They developed a convolutionary algorithm for balancing and sequencing an assembly line. Matanachai and Yano [16] developed a heuristic based on filtered beam search for balancing the mixed-model assembly line to reduce work overload as well as to maintain reasonable workload balance among the stations. Jin and Wu [17] considered the mixed-model assembly line balancing problem and developed a heuristic called “variance algorithm” to balance the assembly line. Hop [18] formulated a fuzzy binary linear programming model for the mixed-model assembly line balancing problem. Then it is transformed into a mixed zero-one programming model. Also, the author developed a fuzzy heuristic to minimize the number of workstations. Rahimi-Vased and Mirzaei [19] developed a hybrid multi-objective shuffled frog-leaping algorithm for a mixed-model assembly line sequencing problem. The objective includes minimization of total utility work, total production rate variation and total setup costs. Kilincci [20] developed a new heuristic based on Petri net approach to minimize the number of workstations of the mixed-model assembly line balancing problem. This algorithm makes an order of firing sequence of transactions from Petri net model of precedence diagram. The task is assigned to a workstation using this order and backward procedure. Through an experiment, the author showed that the proposed algorithm performs better. Al-Mamun and Chowdhury [21] and Al-Mamun *et al.* [22] proposed a heuristic to design the mixed-model assembly line with parallel workstations, zoning constraint and resource limitations such that the number of workstations is minimized. Then, they applied this heuristic to solve a case of a plastic bag manufacturing company. Zhang and Han [23] developed an improved differential evolution algorithm for the mixed-model assembly line balancing problem applied to a car manufacturing industry. Fathi *et al.* [24] have developed a new heuristic method based on CPM for simple assembly line balancing problem. Fattahi *et al.* [25] developed a mixed integer programming model to minimize the total number of workers on the line as the first objective and the number of opened multi-manned workstations as the second objective for the multi-manned assembly line balancing problem. Since, this problem is a combinatorial problem, they developed a heuristic based on ant colony optimization approach to solve large size problems.

Rahimi-Vahed *et al.* [26] developed a mathematical model for the mixed-model assembly line balancing problem in which the objectives are simultaneously minimizing total utility work, total production rate variation and total setup cost. Since, it is NP-hard, they developed a new multi-objective scatter search (MOSS) to search locally Pareto-optimal frontier for the problem. They compared the results of the proposed algorithm with that of three prominent multi-objective algorithms, viz. PS-NC GA, NSGA-II and SPEA-II and concluded that MOSS outperforms the existing algorithms.

Normally heuristics take very little time to solve any combinatorial problem, in specific the assembly line balancing problem, but the accuracy of the solution with reference to the optimal solution may not be high. During seventies and eighties, most of the researchers used heuristics to solve assembly line balancing problems. Later, researchers resorted to develop meta-heuristics such as simulated annealing algorithm, genetic algorithm, ant colony optimization algorithm, etc. for the assembly line balancing problems, which are presented in the

following sections.

2.4. Genetic Algorithms

Chutima and Iammi [27] developed a genetic algorithm for the mixed-model assembly line balancing problem in which the objectives are minimizing the number of workstations and total idle time. They compared the proposed algorithm with COMSOAL and reported that the proposed genetic algorithm outperforms COMSOAL. Since, the performance of any genetic algorithm is proved to be superior to that of heuristic like COMSOAL, the comparison should have been done with some powerful heuristics like ACO algorithm, simulated annealing algorithm, etc. Haq *et al.* [28] developed a hybrid genetic algorithm to the mixed-model assembly line balancing problem in which the objective is to minimize the number of workstations for a given cycle time. Su and Lu [29] considered the mixed-model assembly line balancing problem in which the objective is to design the assembly line to smooth the workload balance within each workstation. They developed a genetic algorithm to find the sequence of the models, which will minimize the cycle time. They carried out a simulation experiment. Bai *et al.* [30] considered the mixed-model assembly line balancing problem for which they developed a new hybrid genetic algorithm to find good solution of the problems. Al-e-Hashem and Aryanezhad [31] considered the mixed-model assembly line balancing problem with a bypass sub-line, which processes a portion of assembly operations of products with relatively longer assembly times. They developed a hybrid algorithm based on a genetic algorithm to level the part usage rates and reduce line stoppages. They compared the performance of this algorithm with that of a complete enumeration method and found that the proposed algorithm compares well with the optimal solution. Moon *et al.* [32] have considered the assembly line balancing problem with the objective of minimizing the total annual workstation costs and annual salary of the assigned workers for a given cycle time. In this problem, the workers with variety of skills are assumed. They developed a mixed integer liner programming model with a genetic algorithm for this integrated assembly line balancing problem with resource restrictions. They reported that the proposed algorithm gives efficient solution based on experimentation with numerical problems.

Akpinar and Bayhan [33] developed a hybrid genetic algorithm to minimize the number of workstations, maximize the workload smoothness between workstations and maximize the workload smoothness within workstations of the mixed-model assembly line balancing problem with parallel workstations and zoning constraints. They hybridized using three well known heuristics, Kilbridge & Wester heuristic, Phase-I of Moodie & Yound method and Ranked Positional Weight method with genetic algorithm. They compared the performance of this algorithm with that of existing algorithms using 20 representative problems. They developed a model with single objective and then three goals which are relevant to the problem are incorporated into that single model. Mamun *et al.* [34] developed a genetic algorithm to solve mixed-model assembly line balancing problem with the objective of minimizing the number of workstations. The task times are assumed to be deterministic. Also, they developed a local search heuristic to avoid convergence of optimum solution. This algorithm is applied to a company manufacturing plastic bag and its results are reported. They did not compare the performance of this algorithm with that of the best existing algorithm.

Sivasankaran and Shahabudeen [1] developed a genetic algorithm for the mixed-model assembly line balancing problem with the objective of maximizing the balancing efficiency for a common cycle time which is computed based on the cycle times of the models. In the past researchers used average task time for each of the tasks of the combined model of the mixed-model assembly line balancing problem. But, they used the original task times of the individual models while forming the workstations, which is a realistic approach. They used the cyclic crossover method in the genetic algorithm designed to solve the mixed-model assembly line balancing problem, which is similar to the crossover method used by Senthilkumar and Shahabudeen [35] in the genetic algorithm to minimize the makespan of the open shop scheduling problem. Further, they compared the balancing efficiency of a sample problem using the average task times for the combined model with that using individual task times for the combined model and found that the balancing efficiency using the individual task times for the combined model gives the best solution. Sivasankaran and Shahabudden [36] developed four different generating algorithms by varying crossover methods for single model assembly line balancing problem to maximize the balancing efficiency. They compared the algorithms in terms of balancing efficiency and found that the Algorithm 4 performs better than the other algorithms.

There is a possibility that the performance of a genetic algorithm will be affected by the method of crossover

method used in it. Hence, there is a necessity to analyze the effect of "crossover method" on the balancing efficiency of the mixed-model assembly line balancing problem, while using genetic algorithm to solve it.

2.5. Simulated Annealing Algorithm

Fattahi and Salehi [37] considered the mixed-model assembly line to minimize the total utility and idle costs with variable launching interval for sequencing the models with a deterministic cycle time. They developed a hybrid meta-heuristic based on a simulated annealing to solve the launching interval problem for each sequence. Ozcan *et al.* [38] considered the balancing and sequencing of parallel mixed assembly lines in which two or more assembly lines are balanced together. They developed a simulated annealing algorithm for this problem to minimize the number of workstations and attain equalization of workstations among workstations.

The performance of simulated annealing algorithm applied to the mixed-model assembly line balancing problem will be affected by the seed generation algorithm that is used and the parameters of the simulated annealing algorithm, viz. temperature, reduction factor, etc. So, researchers can devise improved simulated annealing algorithms with efficient seed generation algorithms.

2.6. ACO Algorithm

Yagmahan [39] developed a multi-objective ant colony optimization algorithm to balance the mixed-model assembly line with the objective of minimizing the number of workstations and workload smoothness.

From these literatures, it is observed that the researchers used mathematical models, branch and bound method, heuristics, genetic algorithms, simulated annealing algorithm, ant colony optimization algorithm, etc. to design the assembly line of the mixed-model assembly line balancing problem. In all the heuristics as well as meta-heuristics, a combined model of the models is derived and the average time of each task in the combined model is computed. Then the design of the assembly line is done for a given cycle time, usually the average cycle time, which is common to all the models. The allocation of the tasks to different workstations of each model is carried out based on the tasks of the combined model and the average task times as already explained. Sivasankaran and Shahabudeen [1] *brought out the fact that the use of average task times in the combined model is considered to be unrealistic. Hence, they designed a genetic algorithm for the mixed-model assembly line balancing problem with cyclic crossover method, based on the original task times of the models to maximize the balancing efficiency, which is more realistic. As stated earlier, the performance of genetic algorithm may be affected by crossover method that is used in it. Hence, in this paper, an attempt has been made to compare the performance of the genetic algorithms with different crossover methods applied to the mixed-model assembly line balancing problem in terms of balancing efficiency. So, two crossover methods are proposed to devise two different genetic algorithms. Further, another genetic algorithm is developed, which uses cyclic crossover method and improved method of assigning tasks to workstations. The performance of these three genetic algorithms and the genetic algorithm developed by Sivasankan and Shahabudeen* [1] *are compared using a complete factorial experiment with three factors, viz. Problem Size, Algorithm and Cycle Time.*

3. Problem Statement

The problem considered in this paper is the mixed-model straight type assembly line balancing problem which consists of M models. Each model consists of a set of tasks and the union of the tasks of the models forms the total set of tasks of the combined model. The total number of tasks in the combined model is n. In each model, if a task is present, it has an associated deterministic task time. If a particular task is absent in a model, its time is assumed to be 0. The cycle time is computed based on a given production volume per shift using the following formula [40].

$$\text{Cycle Time}\left(CT\right) = \text{Effective time available per shift} \big/ \text{Production volume per shift}$$

The balancing efficiency of the solution of the line balancing problem is given by the following formula [40].

$$\text{Balancing efficiency} = \left[\frac{\text{Sum of all task times}}{\text{Number of workstations} \times \text{Cycle time}}\right] \times 100 = \left[\left(\sum_{j=1}^{N} t_j\right) \Big/ \left(NS \times CT\right)\right] \times 100$$

where,

N is the number of tasks
t_j is the required time of the task j
CT is the cycle time
NS is the number of workstations

In this paper, the *ALBP* 1 problem with a mixed-model is considered. In this problem, there will be many models which are to be produced in batches using the same assembly line. The presence of a mixed-model makes the design of the assembly line more complex, in terms of processing times of the tasks and cycle times of the model. The average processing time (T_j) of each task as given by the following formula is normally taken as the representative value of the task time for that task.

$$T_j = \left[\left(\sum_{i=1}^{M} t_{ij}\right) \Big/ NM_j\right] \quad \text{for} \quad J = 1, 2, 3, \cdots, n \text{ , where } t_{ij} \neq 0]$$

where,

t_{ij} is the time of the task j in the model i
n is the number of tasks
M is the number of models
T_j is the average time of the task j
NM_j is the number of models in which the processing time of the task j is more than zero.

In this paper, instead of using the average task times for the tasks in the combined model, the original task times of the individual models are used to design the assembly lines, which is a realistic approach.

The objective of this research is to group the tasks of the combined model into a minimum number of workstations without violating precedence constraints for a common cycle time, which is derived based on the cycle times of the models. Generally, the common cycle time is the average of the cycle times of the models. In this paper, three different genetic algorithms are designed for the mixed-model assembly line balancing problem, in which the original times of the tasks are used while concurrently designing the workstations of the models, such that the combined balancing efficiency is maximized. The combined balancing efficiency is the average of the balancing efficiencies of the models. Then, these three algorithms and the genetic algorithm developed by Sivasankaran and Shahabudeen [1] *are compared using a randomly generated data set as per a complete factorial experiment with three factors, viz. Problem Size, Algorithm and Cycle Time, in terms of the combined balancing efficiency.*

4. Design of Genetic Algorithms

This section presents four different genetic algorithms as listed below to design the mixed-model assembly line balancing problem.

- Genetic algorithm with cyclic crossover method, ALG1(Sivasankaran and Shahabudeen [1])
- Genetic algorithm with forward crossover method, ALG2 (Proposed)
- Genetic algorithm with reverse crossover method, ALG3 (Proposed)
- Genetic algorithm with cyclic crossover method and modified workstation formation , ALG4 (Proposed)

The genetic algorithm creates an initial population consisting of say N chromosomes. Each chromosome consists of the task numbers of the combined model in random order from left to right. It then evaluates each chromosome in terms of a fitness function value [41]. In this paper, it is the balancing efficiency.

Next, the chromosomes are arranged in a sorted order based on the balancing efficiency from high to low. Then a percentage of chromosomes will be treated as a subpopulation. The process of crossover and mutation will be carried in it for different pairs of chromosomes to create corresponding offspring. Later, the modified chromosomes (offspring) will replace the corresponding chromosomes in the population. Again, the chromosomes in the population will be sorted based on the balancing efficiency from high to low. The whole process will be repeated for a specified number of iterations and then the best chromosome will be selected as the final solution for implementation.

4.1. Crossover Methods

The different crossover methods are explained in the following subsections.

Consider the combined precedence network of assembling the Model 1 and Model 2, which has ten tasks as

shown in the **Figure 3**. Each chromosome in the original population is generated by randomly assigning the tasks to different gene positions in that chromosome. Two sample chromosomes for the tasks in the **Figure 3** are shown in **Table 1** and **Table 2**.

4.1.1. Cyclic Crossover Method

In this paper, the cyclic crossover method used for creating two offspring, which has been implemented in the genetic algorithm developed by Senthilkumar and Shahabudeen [35] for open shop scheduling, is adapted to create offspring of the chromosomes of the mixed-model assembly line balancing problem. Using chromosome 1 and the chromosome 2, which are shown in **Table 1** and **Table 2**, respectively, the cyclic crossover method is explained below.

Step 1: Generate two crossover points randomly. Let them be 3 and 7.

Step 2: Identify the values of the genes in the Chromosome 1 between the two crossover points inclusive of the crossover points. These values are 8, 3, 4, 1 and 9.

Step 3: Form the offspring 2 whose gene positions are filled with the genes of the chromosome 2 as shown in **Table 3**.

Step 4: In the offspring 2, wherever the gene value is equal to each of the gene values identified between the two crossover points of the chromosome 1, set it to 0 as shown in **Table 4**.

Step 5: Read the non-zero values of the genes from left to right of the offspring 2 and write them in cyclic order starting from the next position of the second crossover point, except the points between the two crossover points as shown in **Table 5**.

Step 6: Write the values of the genes between the two crossover points of the chromosome 1 in the empty positions of the offspring 2 starting from the first vacant position from left as shown in **Table 6** to get the final offspring 2.

Similarly, the offspring 1 to replace the chromosome 1 is created using the cyclic crossover method and it is as shown in **Table 7**.

Table 1. Chromosome 1.

Position	1	2	3	4	5	6	7	8	9	10
Chromosome 1	5	7	8	3	4	1	9	2	10	6

Table 2. Chromosome 2.

Position	1	2	3	4	5	6	7	8	9	10
Chromosome 2	8	1	4	2	5	10	9	6	7	3

Table 3. Offspring 2 with initial genes.

Position	1	2	3	4	5	6	7	8	9	10
Offspring 2	8	1	4	2	5	10	9	6	7	3

Table 4. Modified partial offspring 2.

Position	1	2	3	4	5	6	7	8	9	10
Offspring 2	0	0	0	2	5	10	0	6	7	0

Table 5. Rearranged partial offspring 2.

Position	1	2	3	4	5	6	7	8	9	10
Offspring 2	6	7						2	5	10

Table 6. Final offspring 2 using cyclic crossover method.

Position	1	2	3	4	5	6	7	8	9	10
Offspring 2	6	7	8	3	4	1	9	2	5	10

Table 7. Offspring 1 using cyclic crossover method.

Position	1	2	3	4	5	6	7	8	9	10
Offspring 1	1	6	4	2	5	10	9	7	8	3

4.1.2. Forward Crossover Method

The steps of the forward crossover method proposed in this paper are presented below. The steps 1 to 4 of this method are same as that of the cyclic crossover method and the remaining steps are as follows.

Step 5: Read the non-zero values of the genes from left to right of the offspring 2 in **Table 4** and write them in forward direction starting from the first available gene position from left to right, except the points between the two crossover points as shown in **Table 8**.

Step 6: Write the values of the genes between the two crossover points of the chromosome 1 in the empty positions of the offspring 2 starting from the first vacant position from left as shown in **Table 9** to get the final offspring 2.

Similarly, the offspring 1 to replace the chromosome 1 is created using the forward crossover method and it is as shown in **Table 10**.

4.1.3. Reverse Crossover Method

The steps of the reverse crossover method proposed in this paper are presented in this section. The steps 1 to 4 of this method are same as that of the cyclic crossover method and the remaining steps are as follows.

Step 5: Read the non-zero values of the genes from left to right of the offspring 2 in **Table 4** and write them in reverse direction starting from the last available gene position from right to left, except the points between the two crossover points as shown in **Table 11**.

Step 6: Write the values of the genes between the two crossover points of the chromosome 1 in the empty positions of the offspring 2 starting from the first vacant position from left as shown in **Table 12** to get the final offspring 2.

Similarly, the offspring 1 to replace the chromosome 1 is created using the reverse crossover method and it is shown in **Table 13**.

4.2. Construction of Ordered Vector

The genes of each chromosome are to be ordered (rearranged) such that the serial assignment of the genes from the ordered vector to workstations does not violate the precedence constraints as shown in the **Figure 1** as well as in the **Figure 2**. The steps of constructing the ordered vector for a given chromosome/ offspring as presented by Sivasankaran and Shahabudeen [1] are given below.

Step 1: Input the chromosome.

Let the chromosome I, which is already shown in the **Table 1** be as shown in **Table 14** along with the values for the STATUS row as zero. If the value of the STATUS for a gene (task) is zero, then it signifies that it is not assigned to any workstation; otherwise, it signifies that it is assigned to some workstation.

- Form the immediate predecessor(s) matrix of the tasks shown in the **Figure 3** as shown in **Table 15**. The maximum of the number of immediate predecessors of the models in the **Figure 3** is 3.
- Number of tasks (genes of the chromosome), *N*
- Initialize the gene position of the ordered vector, K = 1
- Set the chromosome number I

Step 2: Set the gene position of the chromosome, J = 1

Step 3: If the STATUS of the gene J of the chromosome I is equal to 1, then go to Step 9; else, go to Step 4.

Step 4: If all the values in the row of the **Table 15** corresponding to the task at the gene position J of the chromosome I are zero, then go to Step 5; otherwise, go to Step 9.

Table 8. Rearranged partial offspring 2.

Position	1	2	3	4	5	6	7	8	9	10
Offspring 2	2	5						10	6	7

Table 9. Final offspring 2 using forward crossover method.

Position	1	2	3	4	5	6	7	8	9	10
Offspring 2	2	5	8	3	4	1	9	10	6	7

Table 10. Offspring 1 using forward crossover method.

Position	1	2	3	4	5	6	7	8	9	10
Offspring 1	7	8	4	2	5	10	9	3	1	6

Table 11. Rearranged partial offspring 2.

Position	1	2	3	4	5	6	7	8	9	10
Offspring 2	7	6						10	5	2

Table 12. Final offspring 2 using reverse crossover method.

Position	1	2	3	4	5	6	7	8	9	10
Offspring 2	7	6	8	3	4	1	9	10	5	2

Table 13. Offspring 1 using reverse crossover method.

Position	1	2	3	4	5	6	7	8	9	10
Offspring 1	6	1	4	2	5	10	9	3	8	7

Table 14. Chromosome I.

Gene Position J	1	2	3	4	5	6	7	8	9	10
STATUS	0	0	0	0	0	0	0	0	0	0
Chromosome 1	5	7	8	3	4	1	9	2	10	6

Table 15. Immediate predecessors matrix [IP_{pq}].

Task (p)	Immediate Predecessor (s) (q)
1	Nil
2	1
3	1
4	1
5	4
6	2
7	3 4
8	5
9	8
10	6 7 9

Step 5: Assign the task at the gene position J of the chromosome I to the gene position K of the ordered vector.

$OV_K = C_{IJ}$

Step 6: Set the status of the gene position J of the chromosome I to 1 in **Table 14**. $STATUS_J = 1$

Step 7: Change the value of C_{IJ} in immediate predecessor matrix to zero, wherever it is equal to the task at the gene position J.

Step 8: Increment the gene position (K) of the ordered vector by 1 and go to Step 2.

Step 9: Increment the gene position (J) of the chromosome 1 by 1.

Step 10: If $J \leq N$, then go to Step 3; otherwise go to Step 11.

Step 11: Stop.

The application of the above steps to the chromosome 1 which is shown in **Table 1** gives an ordered vector as shown in **Table 16**.

4.3. Evaluation of Fitness Function of Chromosome

The cycle time of the model 1as well as that of the model 2 is assumed as 20 minutes. Hence, the cycle time of the combined model is also 20 minutes, which is the average of the cycle times of both the models. The fitness function of the chromosome 1, namely balancing efficiency is obtained by assigning the tasks serially from left to right from its ordered vector 1-3-4-5-7-8-9-2-6-10 into workstations for the given cycle time of 20 minutes of the combined model as shown in **Table 17**.

While assigning a task into a workstation, that task pertaining to all the models should be assigned to the same workstation. If a task is available in only one model then that can be independently assigned to the current workstation.

Table 16. Ordered vector of chromosome 1 shown in Table 1.

Gene Position J	1	2	3	4	5	6	7	8	9	10
Chromosome 1	1	3	4	5	7	8	9	2	6	10

Table 17. Solution and fitness function value for ordered vector 1-3-4-5-7-8-9-2-6-10.

Workstation	Assigned Tasks		Unassigned Time	
	Model 1	Model 2	Model 1	Model 2
I	1	1	13	12
	3	3	**4**	**2**
II	4	4	10	6
	-	5	**10**	**0**
III	7	7	12	15
	-	8	12	9
	-	9	**12**	**1**
IV	2	-	8	20
	6	-	2	20
V	10	10	7	8
Balancing efficiency			65%	86.25%
Combined efficiency				75.63%

4.4. Genetic Algorithm with Cyclic Crossover Method (ALG_1)

The steps of *Genetic Algorithm with Cyclic Crossover Method* to group the tasks of the mixed-model assembly line balancing problem into a minimum number of workstations developed by Sivasankaran and Shahabudeen [1] are presented below. Here, the balancing efficiencies of the individual models and the combined balancing efficiency, which is the average of the balancing efficiencies of the individual models, are computed. The objective of the design is to find the solution with the minimum number of workstations such that the combined balancing efficiency is maximized.

Step 1: Input the following.

Number of models, M

Number of tasks of the combined models, n

Processing times of tasks, T_{ij}, $i = 1, 2, 3, \cdots, M$ and $j = 1, 2, 3, \cdots, n$

(If the task j is not available in the model i, then $T_{ij} = 0$)

Number of immediate predecessor(s) of the task j of the combined model, NIP_j, $j = 1, 2, \cdots, n$

Immediate predecessor(s) matrix of the combined model, IP_{jq}, $j = 1, 2, \cdots, n$, $q = 1, 2, \cdots, NIP_p$

Average (common) cycle time, CT

Mutation probability, α (0.3).

Best Balancing Efficiency, $BBE = 0$

Step 2: Set the genetic algorithm parameters as given below.

- Size of population, N
- Size of subpopulation, m (30% of N). It should be rounded to the next even integer.
- Number of iterations to be carried out, Q

**Creation and evaluation of population*

Step 3: Construct N chromosomes of the population by randomly assigning the tasks to different gene positions in each of the chromosomes, $C_{K,J}$, $K = 1, 2, \cdots, N$ and $J = 1, 2, \cdots, n$.

Step 4: For each chromosome, $K = 1, 2, \cdots, N$, perform the following.

4.1. Find ordered vector of each chromosome (OV_J, $J = 1, 2, 3, \cdots, n$)

4.2. Design workstation for the given cycle time using the processing times of the tasks of the individual models and the combined immediate predecessor(s) matrix [$IP_{j,q}$, $j = 1, 2, 3, \cdots, n$, $q = 1, 2, 3, \cdots, NIP_q$]

4.3. Find the sum of task times of the model I (STT_I), for $I = 1, 2, 3, \cdots, M$

4.4. Find the balancing efficiency of each model I (η_I) using the following formula.

$$\eta_I = \left[SST_I / (m \times CT) \right] \times 100,\ I = 1, 2, 3, \cdots, M$$

where, m is the number of workstations to which the tasks of the model I are assigned.

4.5. Find the combined balancing efficiency, which is the average of the balancing efficiencies of the models and store it in BE_K.

Step 5: Set the Iteration Number q to 1.

**Sorting the population*

Step 6: Sort the chromosomes in the descending order of their balancing efficiencies

Let the sorted chromosomes be, $SC_{K,J}$, $K = 1, 2, \cdots, N$, $J = 1, 2, \cdots, n$ and the array of their Sorted Balancing Efficiency be SBE_K, $K = 1, 2, \cdots, N$.

Step 7: Copy the sorted chromosomes, $SC_{K,J}$, $K = 1, 2, \cdots, N$, $J = 1, 2, \cdots, n$ into $C_{K,J}$, $K = 1, 2, \cdots, N$, $J = 1, 2, \cdots, n$ along with their sorted balancing efficiencies SBE_K, $K = 1, 2, \cdots, N$ into balancing efficiency array, BE_K, $K = 1, 2, \cdots, N$.

Step 8: Perform the following steps to update the balancing efficiency if applicable.

8.1 If $BE_1 \le BBE$, then go to Step 9; otherwise go to Step 8.2.

8.2 Update the following.

Best balancing efficiency, $BBE = BE_1$

Best Chromosome, $BC_J = SC_{1,J}$, $J = 1, 2, \cdots, n$.

Step 9: If $q > Q$, then go to Step 20; otherwise, go to Step 10.

**Workings on Subpopulation*

Step 10: Treat the topmost 30% of the population ($0.3N = P$) of the new population after sorting as the sub-

population for the crossover operation. If the size of the subpopulation is not an even number, then add 1 to the size of the subpopulation.

Step 11: Set chromosome number, $P = 1$

Step 12: Perform two point cyclic crossover between the chromosomes P and P+1 as listed below and obtain the offspring whose numbers are P and P+1.

Crossover between: $C_{P,J}$, $J = 1, 2, \cdots, n$ & $C_{P+1,J}$, $J = 1, 2, \cdots, n$

Offspring obtained: $OS_{P,J}$, $J = 1, 2, \cdots, n$ & $OS_{P+1,J}$, $J = 1, 2, \cdots, n$

Step 13: Perform mutation in each of the offspring for a mutation probability of α.

Step 14: For each of the offspring [offspring P and offspring P+1], perform the following.

14.1 Find the ordered vector of the offspring

14.2 Design the workstation for the given cycle time using Step 4.2 and find the combined balancing efficiencies BE_K and BE_{K+1}.

Step 15: Increment chromosome number by 2, $P = P + 2$.

Step 16: If $P \leq m$ then go to Step 12; otherwise, go to Step 17.

Step 17: Copy the new offspring $OS_{R,J}$, $R = 1, 2, \cdots, M$ and $J = 1, 2, \cdots, n$ to the respective chromosome vectors, $C_{P,J}$, $C = 1, 2, \cdots, M$ and $J = 1, 2, \cdots, n$, respectively.

Step 18: Increment the iteration number by 1 ($q = q + 1$).

Step 19: Go to Step 6.

Step 20: Perform the following.

Find the ordered vector of the best chromosome, BC_J, $J = 1, 2, \cdots, n$.

Design the workstation for the given cycle time using Step 4.2 and print the following.

Station details of the models

Best combined balancing efficiency *BBE*

Step 21: Stop.

4.5. Genetic Algorithm with Forward Crossover Method for Mixed-Model Assembly Line Balancing Problem (ALG_2)

The steps of the *genetic algorithm with the forward crossover method* proposed in this paper for the mixed-model assembly line balancing problem are same as that of ALG_1, except the Step 12. The required Step 12 of this algorithm is given below.

Step 12: Perform two point *forward crossover* between the chromosomes P and $P + 1$ as listed below and obtain the offspring whose numbers are P and P+1.

Crossover between: $C_{P,J}$, $J = 1, 2, \cdots, n$ & $C_{P+1,J}$, $J = 1, 2, \cdots, n$

Offspring obtained: $OS_{P,J}$, $J = 1, 2, \cdots, n$ & $OS_{P+1,J}$, $J = 1, 2, \cdots, n$

4.6. Genetic Algorithm with Reverse Crossover Method for Mixed-Model Assembly Line Balancing Problem (ALG_3)

The steps of the *genetic algorithm with reverse crossover method* proposed in this paper are same as that of ALG_1, except the Step 12. The required Step 12 of this algorithm is presented below.

Step 12: Perform two point *reverse crossover* between the chromosomes P and P+1 as listed below and obtain the offspring whose numbers are P and P+1.

Crossover between: $C_{P,J}$, $J = 1, 2, \cdots, n$ & $C_{P+1,J}$, $J = 1, 2, \cdots, n$

Offspring obtained: $OS_{P,J}$, $J = 1, 2, \cdots, n$ & $OS_{P+1,J}$, $J = 1, 2, \cdots, n$

4.7. Genetic Algorithm with Cyclic Crossover Method and Modified Workstation Formation for Mixed-Model Assembly Line Balancing Problem (ALG_4)

The steps of the *genetic algorithm with cyclic crossover method and modified workstation formation* proposed in this paper are same as that of ALG_1, except the Step 4.2. The required Step 4.2 of this algorithm is shown below.

Step 4.2 Design the workstation for the given cycle time using the processing times of the individual models and the immediate predecessor(s) matrix of the combined model as per the ordered vector with the following addition.

While moving from one workstation to another workstation, if there is idle time in the current workstation, then look for alternate succeeding task(s) which can best fit into the current workstation that will result with either zero idle time or least idle time in that workstation.

5. Comparison of Genetic Algorithms

This section presents the comparison of the four genetic algorithms, viz. ALG_1, ALG_2, ALG_3 and ALG_4 using a complete factorial experiment with three factors, viz. Problem Size (A), Algorithm (B) and Cycle Time (C). The levels of the Problem Size (A) vary from 40 to 65 in steps of 5. The levels of Algorithm (B) are ALG_1, ALG_2, ALG_3 and ALG_4. For each problem, the Cycle Time (C) is set at two levels, viz.75 min and 100 min. The number of replications under each experimental combination of the three factors is 2. The total number of observations of this experiment is 96. The ANOVA model of this complete factorial experiment [42] [43] is shown below.

$$Y_{ijkl} = \mu + A_i + B_j + AB_{ij} + C_k + AC_{ik} + BC_{jk} + ABC_{ijk} + e_{ijkl}$$

where,

μ is the overall mean

Y_{ijkl} is the combined balancing efficiency of the l^{th} replication under the i^{th} level of the factor A, j^{th} level of the factor B and the k^{th} level of the factor C.

A_i is the effect of the i^{th} level of the factor A on the combined balancing efficiency

B_j is the effect of the j^{th} level of the factor B on the combined balancing efficiency

AB_{ij} is the effect of the i^{th} level of the factor A and the j^{th} level of the factor B on the combined balancing efficiency

C_k is the effect of the k^{th} level of the factor C on the combined balancing efficiency

AC_{ik} is the effect of the i^{th} level of the factor A and the k^{th} level of the factor C on the combined balancing efficiency

BC_{jk} is the effect of the j^{th} level of the factor B and the k^{th} level of the factor C on the combined balancing efficiency

ABC_{ijk} is the effect of the i^{th} level of the factor A, j^{th} level of the factor B and the k^{th} level of the factor C on the combined balancing efficiency

e_{ijkl} is the error associated with the combined balancing efficiency of the l^{th} replication under the i^{th} level of the factor A, j^{th} level of the factor B and the k^{th} level of the factor C.

The number of models considered in all the replications is 2. The combined mean balancing efficiencies of the replications under all the experimental combinations as per the design of the complete factorial experiment are given in **Table 18**.

Table 18. Combined mean balancing efficiencies of algorithms as per complete factorial experiment.

Problem size (A)	Replication	Algorithm (Crossover Method) [B]							
		CYCLIC (ALG_1)		FORWARD (ALG_2)		REVERSE (ALG_3)		CYCLIC_MODIFIED (ALG_4)	
		Cycle Time (C) Minutes		Cycle Time (C) Minutes		Cycle Time(C) Minutes		Cycle Time (C) Minutes	
		75	100	75	100	75	100	75	100
40	1	73.57	77.25	73.57	77.25	73.57	77.25	73.57	77.25
	2	73.05	85.22	73.05	76.70	73.05	76.70	68.18	68.18
45	1	63.33	65.77	63.33	71.25	63.33	65.77	76.00	71.25
	2	57.00	65.77	57.00	71.25	57.00	65.77	71.25	71.25
50	1	64.96	79.73	75.21	76.32	64.96	79.73	73.08	87.70
	2	69.02	73.30	73.33	80.00	58.67	80.00	73.33	80.00
55	1	61.02	61.91	61.02	61.91	61.02	61.91	77.96	75.18
	2	60.58	65.31	60.58	74.64	60.58	74.64	73.33	80.39
60	1	55.21	55.21	55.21	68.20	55.21	68.21	67.22	77.30
	2	50.81	53.71	68.83	53.71	50.82	53.71	75.02	78.77
65	1	59.87	63.82	59.88	63.81	59.87	68.21	76.98	86.61
	2	54.71	62.63	48.05	62.63	54.71	62.63	79.33	79.33

The different hypotheses of this comparison are listed below.

Factor A (Problem Size:

H_0: There are no significant differences among the treatments of the Factor A in terms of average balancing efficiency of the models.

H_1: There are significant differences among the treatments of the Factor A in terms of average balancing efficiency of the models.

Factor B (Algorithm):

H_0: There are no significant differences among the treatments of the Factor B in terms of average balancing efficiency of the models.

H_1: There are significant differences among the treatments of the Factor B in terms of average balancing efficiency of the models.

Interaction AB (Problem Size x Algorithm):

H_0: There are no significant differences among the treatment combinations of the Factor A and the Factor B in terms of average balancing efficiency of the models.

H_1: There are significant differences among the treatment combinations of the Factor A and the Factor B in terms of average balancing efficiency of the models.

Factor C (Cycle Time):

H_0: There are no significant differences among the treatment of the Factor C in terms of average balancing efficiency of the models.

H_1: There are significant differences among the treatment of the Factor C in terms of average balancing efficiency of the models.

Interaction AC (Problem Size x Cycle Time):

H_0: There are no significant differences among the treatment combinations of the Factor A and the Factor C in terms of average balancing efficiency of the models.

H_1: There are significant differences among the treatment combinations of the Factor A and the Factor C in terms of average balancing efficiency of the models.

Interaction BC (Algorithm x Cycle Time):

H_0: There are no significant differences among the treatment combinations of the Factor B and the Factor C in terms of average balancing efficiency of the models.

H_1: There are significant differences among the treatment combinations of the Factor B and the Factor C in terms of average balancing efficiency of the models.

Interaction ABC (Problem Size x Algorithm x Cycle Time):

H_0: There are no significant differences among the treatment combinations of the Factor A, Factor B and the Factor C in terms of average balancing efficiency of the models.

H_1: There are significant differences among the treatment combinations of the Factor A, Factor B and the Factor C in terms of average balancing efficiency of the models.

The results of ANOVA for the data given in the **Table 18** are summarized in **Table 19**.

Table 19. Results of ANOVA.

Source of variation	Sum of squares	Degrees of freedom	Mean sum of squares	F ratio	F table	Remarks
Problem size A	2227.625	5	445.525	22.833	2.418	**Significant**
Algorithm B	1908.125	3	636.042	32.597	2.808	**Significant**
AB	1352.625	15	90.175	4.621	1.888	**Significant**
Cycle time C	845.000	1	845.000	43.306	4.048	**Significant**
AC	125.531	5	25.106	1.287	2.418	Insignificant
BC	62.594	3	20.865	1.069	2.808	Insignificant
ABC	244.750	15	16.317	0.836	1.888	Insignificant
Error	936.594	48	19.512			
Total	7702.844	95				

From the **Table 19**, it is clear that the components A, B, AB and C are significant and the remaining components are insignificant. Based on these results, the inferences of the model are listed below.

- There are significant differences between problem sizes in terms of balancing efficiency.
- There are significant differences between the algorithms in terms of balancing efficiency.
- There are significant differences between the cycle times in terms of balancing efficiency.
- There are significant differences between the interaction terms of "Problem Size" and "Algorithm in terms" of balancing efficiency.
- There is no significance for the remaining components of the model.

Since, there are significant differences between the algorithms, viz. ALG_1, ALG_2, ALG_3 and ALG_4 in terms of balancing efficiency, the next step is to identify the best algorithm using Duncan's multiple range test.

Determination of the Best Algorithm Using Duncan's Multiple Range Test

As stated above, the best algorithm is identified using Duncan's multiple range test.

Step 1: The ascending order of the mean balancing efficiencies of the algorithms is as shown below. The mean balancing efficiency of an algorithm is the mean of the average balancing efficiencies of the models of all the observations under that algorithm in **Table 18**.

Algorithm	ALG1	ALG3	ALG2	ALG4
Mean balancing efficiency	64.698	65.305	66.869	75.769

Step 2: The standard error of algorithm-mean is computed as shown below.

Std Error = $[MSS_{error}/\text{No. of replications under each algorithm}]^{1/2}$

where, MSS_{error} is as shown in **Table 19**, which is equal to 19.512.

Std. Error = $[19.512/24]^{1/2} = 0.901665$

Step 3: The three (4-1) significant ranges from Duncan's table for error degrees of freedom of 48 at $\alpha = 0.05$ are shown below.

j	2	3	4
Significant range	2.848	2.998	3.092

Step 4: The three least significant ranges (LSR) are obtained by multiplying the respective significant ranges with the standard error and they are as shown below.

j	2	3	4
LSR_j	2.568	2.703	2.788

Step 5: The actual ranges (AR) between different means of the algorithms along with corresponding LSR values are shown in **Figure 5**.

From the **Figure 5**, it is clear that the algorithm ALG4 is significantly different from algorithms ALG1, ALG2 and ALG3 in terms of mean balancing efficiency of algorithm. The actual ranges of the remaining combinations of the algorithms are less than the respective LSR values. This means that there are no significant differences between the corresponding pairs of algorithms (ALG2 and ALG1, ALG2 and ALG3, and ALG3 and ALG1) in terms of mean balancing efficiency of algorithm.

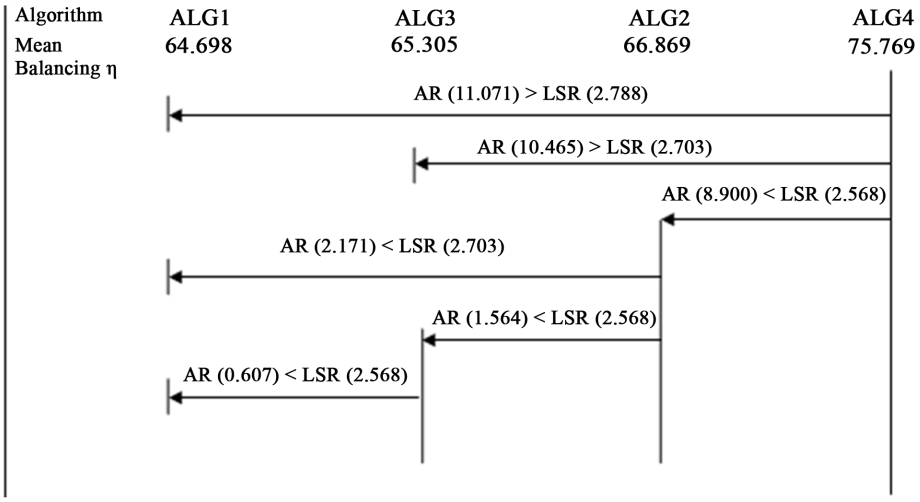

Figure 5. Comparison of actual ranges (AR) with respective lease significant ranges (LSR).

Since the algorithm ALG4 is significantly different and superior to ALG1, ALG2 and ALG3 in terms of the average balancing efficiency of the models, the algorithm ALG4 is identified as the best algorithm to solve this mixed-model assembly line balancing problem.

6. Zero-One Programming Model for Mixed-Model Assembly Line Balancing Problem

The closeness of the solution in terms of balancing efficiency of the mixed-model assembly line balancing problem obtained using the best algorithm ALG4 identified in Section 5 to the corresponding optimal solution can be verified using the result of a mathematical model for the mixed-model assembly line balancing problem.

Hence, in this section the zero-one programming model presented by Gokcen and Erel [4] is presented based on which randomly generated test data are solved to perform comparison. The mathematical model is designed to minimize the number of workstations, which in turn maximizes the balancing efficiency. For each problem, the solution of the mathematical model is converted into balancing efficiency using the formula given in Section 3.

Variables of the Model:

Let,

N be the number of tasks in the problem, which is the union of the tasks in all the models

K be the feasible number of workstations assumed/ obtained using any simple heuristic

M be the number of models

PT_i be the set of preceding tasks for the task i, for $i = 1, 2, 3, \cdots, N$

F_i be the set of following tasks for the task i, for $i = 1, 2, 3, \cdots, N$

T_{ij} be the task time of task i of model j, for $i = 1, 2, 3, \cdots, N$ and $j = 1, 2, 3, \cdots, M$

C be the common cycle time of the models

E_{ij} be the earliest workstation to which the task i of the model j can be assigned as per the precedence network, for $i = 1, 2, 3, \cdots, N$ and $j = 1, 2, 3, \cdots, M$

L_{ij} be the latest workstation to which the task i of the model j can be assigned as per the precedence network, for $i = 1, 2, 3, \cdots, N$ and $j = 1, 2, 3, \cdots, M$

Y_{ik} = 1, if the task i is assigned to workstation k; = 0, otherwise.

for $i = 1, 2, 3, \cdots, N$ and $k = 1, 2, 3, \cdots, K$

X_{jk}= 1, if the model j uses the workstation k; = 0; otherwise.

for $j = 1, 2, 3, \cdots, M$ and $k = 1, 2, 3, \cdots, K$

Z_{jk} be the set of tasks of the model j, which can be assigned to workstation k

SZ_{jk} be the size of the set Z_{jk}, that is the number of tasks in Z_{jk},

for $j = 1, 2, 3, \cdots, M$ and $k = 1, 2, 3, \cdots, K$

A_k = 1, if the all the models use the workstation k (X_{jk} =1 for $j = 1, 2, 3, \cdots, M$))

= 0 (X_{jk} = 0, for $j = 1, 2, 3, \cdots, M$), otherwise

for $k = 1, 2, 3, \cdots, K$

The estimates of E_{ij} and L_{ij} are obtained using the following formulas.

$$E_{ij} = \left[\left(T_{ij} + \sum_{o(-PT_i} T_{oj} \right) \Big/ C \right]^+ \quad \text{for} \quad i = 1, 2, 3, \cdots, N \quad \text{and} \quad j = 1, 2, 3, \cdots, M \quad \text{(Rounded to next integer)}$$

$$L_{ij} = K + 1 - \left[\left(T_{ij} + \sum_{o(-F_i} T_{oj} \right) \Big/ C \right]^+ \quad \text{for} \quad i = 1, 2, 3, \cdots, N \quad \text{and} \quad j = 1, 2, 3, \cdots, M \quad \text{(Rounded to next integer)}$$

E_i be the earliest workstation of the combined network of the models to which the task i can be assigned (Lower limit of the workstation number of the combined network to which the task i can be assigned) which is given by the following formula.

$$E_i = \text{Max}\left(E_{ij} \right) \text{ among all values of } \quad j = 1, 2, 3, \cdots, M$$

L_i be the latest workstation of the combined network of the models to which the task i can be assigned (upper limit of the workstation number of the combined network to which the task i can be assigned) which is given by the following formula.

$$L_i = \text{Max}\left(L_{ij} \right) \text{ among all values of } \quad j = 1, 2, 3, \cdots, M$$

A zero-one programming model to minimize the number of workstations is presented below.

$$\text{Minimize } W = \sum_{k=1}^{K} A_k$$

Subject to

$$\sum_{k(-E_i \text{ to } L_i}^{K} Y_{ik} = 1, \text{ for } i = 1, 2, 3, \cdots, N \tag{1}$$

$$\sum_{k=E_p}^{L_p} kY_{pk} - \sum_{k=E_q}^{L_q} kY_{qk} \le 0 \tag{2}$$

for pair of tasks p and q in the combined precedence network

$$\sum_{i(-Z_{jk}} T_{ij} Y_{ik} \le C, \text{ for } j = 1, 2, 3, \cdots, M \text{ and } k = 1, 2, 3, \cdots, K \tag{3}$$

$$\sum_{i(-Z_{jk}} Y_{ik} - SZ_{jk} X_{jk} \le 0, \text{ for } j = 1, 2, 3, \cdots, M \text{ and } k = 1, 2, 3, \cdots, K \tag{4}$$

$$\sum_{j=1}^{M} X_{jk} - MA_k = 0, \text{ for } k = 1, 2, 3, \cdots, K \tag{5}$$

where,

$Y_{ik} = 1$, if the task i is assigned to workstation k; = 0, otherwise.

for $i = 1, 2, 3, \cdots, N$ and $k = 1, 2, 3, \cdots, K$

$X_{jk} = 1$, if the model j uses the workstation k; = 0; otherwise.

for $j = 1, 2, 3, \cdots, M$ and $k = 1, 2, 3, \cdots, K$

$A_k = 1$, if the all the models use the workstation k;

= 0, otherwise

for $k = 1, 2, 3, \cdots, K$

W is the optimized number of workstations

The objective function minimizes the number of workstations from given K workstations such that each selected workstation is used by all the models. The selected workstation numbers may not be in continuous order (e.g. 1, 2, 3, 5). So, they must be numbered serially later (e.g. 1, 2, 3, 4).

The constraint i in the constraint set 1assigns the task i to only one workstation. *The constraints in the constraint set* 1 *are called as task assignment constraints.*

The constraint with respect to the pair of tasks p and q such that the task p precedes the task q, in the combined precedence network in the constraint set 2, assigns the task p to earlier workstation when compared to the workstation to which the task q is assigned or assigns the tasks p and q to the same workstation to main the precedence relationship among the tasks p and q (p < q)). This actually makes the sum of the workstation numbers from the lower limit to the upper limit for the workstation number with respect to the task p less than or equal to that with respect to the task q. *The constraints in the constraint set* 2 *are called as task precedence constraints.*

The constraint j in the constraint set 3 maintains the relationship that the sum of the times of the tasks of the model j, which are assigned to the workstation k (Z_{jk}) is less than or equal to the cycle time of the model j. *The constraints in the constraint set* 3 *are called as workstation constraints.*

The constraint with respect to the workstation k and the model j in the constraint set 4 ensures that at least one task of the model j from the subset Z_{jk} is assigned to the workstation k if the workstation k is used by the model j; otherwise, no task of the model j from the subset Z_{jk} is assigned to the workstation k. The constraint k in the constraint set 5 ensures that if the workstation k is selected for assigning tasks, then all the models use that workstation. *The constraint set* 4 *and the constraint set* 5 *put together called as workstation constraints*, which mainly ensues that the number of workstations is same the same for all the models. If a workstation is not used by a model, then all other models will not use that workstation.

7. Comparison of Results of Alg_4 with Results of Zero-One Programming Model

In the section 5, it is concluded that the algorithm ALG4 performs better when compared to the other algorithms, viz. ALG1, ALG2 and ALG3. In this section, the results of the algorithm ALG4 are compared with those of the mathematical model presented in the section 6. The objective of this comparison is to check the closeness of the solutions obtained using the algorithm ALG4 with the corresponding optimal solutions obtained using the mathematical model. Hence, in this section, four different problem sizes (number of tasks), viz. 10, 15, 20 and 25

are considered with two replications in each problem size. The number of models in each of these problem sizes is assumed to be 2. The problem size represents the number of tasks in the combined network of all the models. A common cycle time of 20 minutes is assumed for the replications of the problem with 10 tasks and a common cycle time of 30 minutes is assumed for the replications of all other problems.

Each replication of each problem size considered is solved using the algorithm ALG4 and the corresponding result in terms of average balancing efficiency of the models is shown in **Table 20**. Then, a zero-one programming model is developed for each problem to minimize the number of workstations. The number of workstations obtained using the algorithm ALG4 is assumed as the value of m in the mathematical model of the respective problem. The objective function of the model minimizes the number of workstations. So, the output of the model will be the minimized number of workstations. This is converted into the corresponding balancing efficiencies of the individual models. Based on these efficiencies, the average balancing efficiency of the models is computed. The results of all the problems using the zero-one programming model solved using LINGO 14.0 are show in the **Table 20**. This software uses branch and bound method [44]. From **Table 20**, it is clear that the algorithm ALG4 gives optimal solution for all the problems.

From **Table 20**, it is clear that for the problem with respect to each replication of each problem size, the balancing efficiencies of Model 1, Model 2 and the average balancing efficiencies of the Model 1 and Model 2 obtained using the zero-one programming model and those obtained using the algorithm ALG4, respectively, are the same except for the problem size with 45 nodes. From **Table 20**, it is clear that the mathematical model of the problem with 45 tasks in the combined network of the mixed-model assembly line balancing problem (replication 1 of problem size 45) takes 7 hr 30 min 45 sec. So, the mathematical models of problems of higher sizes cannot be solved using LINGO software.

The inference that the algorithm ALG4 gives optimal solution for each of the problems and the inference that the algorithm ALG4 is the best among the four algorithms compared in the Section 5, clearly reveal that the algorithm ALG4 is efficient in terms of giving near optimal solution for large size problems.

8. Conclusions

In this paper, three genetic algorithms as presented below have been developed. Along with these three genetic algorithms, the authors considered the genetic algorithm with cyclic crossover method (ALG_1) developed by Sivasankaran and Shahabudeen [1] for comparison.

- Genetic algorithm with forward crossover method, ALG_2
- Genetic Algorithm with reverse crossover method, ALG_3
- Genetic algorithm with cyclic crossover method and modified workstation formation, ALG_4

Table 20. Results of zero one programming model and algorithm Alg4 for problems.

Problem (No. of Tasks)	Replication	Cycle time	Mathematical Model					Results of Algorithm 4 (ALG4)				
			No. of Variables	No. of Constraints	Results							
					B.E of M1 (%)	B.E of M2 (%)	A.B.E (%)	CPU time Sec.	B.E of M1 (%)	B.E of M2 (%)	A.B.E. (%)	CPU Time (Sec.)
10	1	20	46	48	81.25	86.25	83.75	0.20	81.25	86.25	83.75	3.08
	2	20	40	47	84.00	74.00	79.00	0.05	84.00	74.00	79.00	3.12
15	1	30	76	65	80.83	77.50	79.12	0.14	80.83	77.50	79.12	4.94
	2	30	68	62	66.00	77.33	71.67	0.20	66.00	77.33	71.67	4.89
20	1	30	117	84	84.44	88.89	86.67	0.16	84.44	88.89	86.67	7.41
	2	30	94	76	85.86	81.11	83.33	0.28	85.86	81.11	83.33	7.41
25	1	30	161	92	97.78	92.22	95.00	0.33	97.78	92.22	95.00	11.15
	2	30	153	99	92.78	93.89	93.33	0.75	92.78	93.89	93.33	12.36
45		100	466	188	Even after 7 hr 30 min 45 sec., the mathematical model does not give solution.				86.00	85.00	85.50	27.96

Note: B.E means Balancing Efficiency, A.B.E. means Average Balancing Efficiency, M1 means Model 1 and M2 means Model 2.

These four algorithms are compared using a complete factorial experiment with three factors, viz. Problem Size (A), Algorithm (B) and Cycle Time (C). The number of levels of the factor "Problem Size (A)" is 6, viz. 40 tasks, 45 tasks, 50 tasks, 55 tasks, 60 tasks and 65 tasks. The factor "Algorithm (B)" is with four levels, viz. ALG_1, ALG_2, ALG_3 and ALG_4. The cycle time (Factor C) is set at two levels, viz. 75 min and 100 min. The number of replications under each experimental combination of the three factors is 2.

Based on the results of the three factor ANOVA experiment, it is observed that the Factor A (Problem Size), Factor B (Algorithm), Factor C (Cycle Time) and the interaction component AB are having significant effect on the combined mean balancing efficiency of the mixed-model assembly line balancing problem and the other components of the ANONA model do not have effect on the combined mean balancing efficiency of the mixed-model assembly line balancing problem, at a significance level of 0.05. From this observation, it is clear that there is significant difference between the genetic algorithms in terms of the combined mean balancing efficiency.

Since, there is significant difference between the algorithms in terms of combined mean balancing efficiency, the best algorithm is determined using Duncan's multiple range test, which gives ALG4 as the best algorithm to the mixed-model assembly line balancing problem in which the combined mean balancing efficiency is maximized.

The accuracy of the solutions of the algorithm ALG4 can be checked by comparing its solutions with the solutions of a mathematical model applied to this mixed-model assembly line balancing problem with the objective of maximizing the balancing efficiency. Hence, a zero-one programming model for this problem is presented. The performance of the algorithm ALG4 is compared with that of the zero-one programming model using another set of randomly generated data consisting of four problem sizes, viz. 10, 15, 20, 25 with two replications and a problem with 45 nodes with one replication in the combined network. It is found that for each of the problems, its solution using the algorithm ALG4 is same as that using the zero-one programming model, except for the problem with 45 nodes whose solution was not obtained even after 7.5 hours by the model. From this analysis, it is clear that for small and moderate size problems, the algorithm ALG4 gives optimal solution as obtained using the zero-one programming model.

In future research, probabilistic task times for the mixed-model assembly line balancing may be assumed and accordingly the results may be analyzed.

Acknowledgements

The authors thank anonymous referees for their suggestions and comments, which helped to improve the content and presentation of this paper.

References

[1] Sivasankaran, P. and Shahabudeen, P. (2013) Genetic Algorithm for Concurrent Balancing of Mixed-Model Assembly Lines with Original Task Times of Models. *Intelligent Information Management*, **5**, 84-92. http://dx.doi.org/10.4236/iim.2013.53009

[2] Sivasankaran, P. and Shahabudeen, P. (2014) Literature Review of Assembly Line Balancing Problems. *International Journal of Advanced Manufacturing Technology*, **73**, 1665-1694. http://dx.doi.org/10.1007/s00170-014-5944-y

[3] Gokcen, H. and Erel, E. (1997) A Goal Programming Approach to Mixed-Model Assembly Line Balancing Problem. *International Journal of Production Economics*, **48**, 177-185.

[4] Gokcen, H. and Erel, E. (1998) Binary Integer Formulation for Mixed-Model Assembly Line Balancing Problem. *Computers & Industrial Engineering*, **34**, 451-461. http://dx.doi.org/10.1016/S0360-8352(97)00142-3

[5] Choi, G. (2009) A Goal Programming Mixed-Model Line Balancing for Processing Time and Physical Workload. *Computers & Industrial Engineering*, **57**, 395-400. http://dx.doi.org/10.1016/j.cie.2009.01.001

[6] Emde, S., Boysen, N. and Scholl, A. (2010) Balancing Mixed-Model Assembly Lines: A Computational Evaluation of Objectives to Smoothen Workload. *International Journal of Production Research*, **48**, 3173-3191. http://dx.doi.org/10.1080/00207540902810577

[7] Kara, Y., Ozgiiven, C., Seeme, N.Y. and Chang, C.T. (2011) Multi-Objective Approaches to Balance Mixed-Model Assembly Lines for Model Mixes Having Precedence Conflicts and Duplicate Common Tasks. *International Journal of Advanced Manufacturing Technology*, **52**, 725-737. http://dx.doi.org/10.1007/s00170-010-2779-z

[8] Sivasankaran, P. and Shahabudeen, P. (2013) Modelling Hybrid Single Model Assembly Line Balancing Problem. *UDYOG PRAGATI*, **37**, 26-36.

[9] Öztürk, C., Tunal, S., Hnich, B. and Örnek, A. (2013) Balancing and Scheduling of Flexible Mixed-Model Assembly Lines with Parallel Stations. *International Journal of Advanced Manufacturing Technology*, **67**, 2577-2591. http://dx.doi.org/10.1007/s00170-012-4675-1

[10] Seker, S., Ozgürler, M. and Tanyas, M. (2013) A Weighted Multi-Objective Optimization Method for Mixed-Model Assembly Line Problem. *Journal of Applied Mathematics*, **2013**, Article ID: 531056.

[11] Akpinar, S. and Baykasoğlu, A. (2014) Modeling and Solving Mixed-Model Assembly Line Balancing Problem with Setups, Part I: A Mixed Integer Linear Programming Model. *Journal of Manufacturing Systems*, **33**, 177-187. http://dx.doi.org/10.1016/j.jmsy.2013.11.004

[12] Akpinar, S. and Baykasoğlu, A. (2014) Modeling and Solving Mixed-Model Assembly Line Balancing Problem with Setups. Part II: A Multiple Colony Hybrid Bees Algorithm. *Journal of Manufacturing Systems*, **33**, 445-461.

[13] Bukchin, Y. and Rabinowitch, I. (2006) A Branch-and-Bound Based Solution Approach for the Mixed-Model Assembly Line-Balancing Problem for Minimizing Stations and Task Duplication Costs. *European Journal of Operational Research*, **174**, 492-508. http://dx.doi.org/10.1016/j.ejor.2005.01.055

[14] Yang, C.J., Gao, J. and Li, J.L. (2014) Balancing Mixed-Model Assembly Lines with Adjacent Task Duplication. *International Journal of Production Research*, **52**, 7454-7471. http://dx.doi.org/10.1080/00207543.2014.937012

[15] Kim, Y.K. and Kim, J.Y. (2000) A Co-Evolutionary Algorithm for Balancing and Sequencing in Mixed-Model Assembly Lines. *Applied Intelligence*, **13**, 247-258. http://dx.doi.org/10.1023/A:1026568011013

[16] Matanachai, S. and Yano, C.A. (2001) Balancing Mixed-Model Assembly Lines to Reduce Work Overload. *IIE Transactions*, **33**, 29-42. http://dx.doi.org/10.1080/07408170108936804

[17] Jin, M. and Wu, S.D. (2002) A New Heuristic Method for Mixed-Model Assembly Line Balancing Problem. *Computers & Industrial Engineering*, **44**, 159-169. http://dx.doi.org/10.1016/S0360-8352(02)00190-0

[18] Hop, N.V. (2006) A Heuristics Solution for Fuzzy Mixed-Modellling Balancing Problem. *European Journal of Operational Research*, **168**, 798-810. http://dx.doi.org/10.1016/j.ejor.2004.07.029

[19] Rahimi-Vahed, A. and Mirzaei, A.H. (2007) A Hybrid Multi-Objective Shuffled Frog-Leaping Algorithm for a Mixed-Model Assembly Line Sequencing Problem. *Computers & Industrial Engineering*, **53**, 642-666. http://dx.doi.org/10.1016/j.cie.2007.06.007

[20] Kilincci, O. (2011) Firing Sequences backward Algorithm for Simple Assembly Line Balancing Problem of Type 1. *Computers and Industrial Engineering*, **60**, 830-839. http://dx.doi.org/10.1016/j.cie.2011.02.001

[21] Al-Mamun, A. and Chowdhury, M.M. (2011) A Heuristic Approach for Balancing Mixed-Model Assembly Line of Type-I Using Genetic Algorithm. *International Conference on Mechanical*, *Production and Automobile Engineering* (*ICMPAE*'2011), Pattaya, 17-18 December 2011, 259-262.

[22] Al-Mamun, A., Khaled, A.A., Ali, S.M. and Chowdhury, M.M. (2012) A Heuristic Approach for Balancing Mixed-Model Assembly Line of Type I Using Genetic Algorithm. *International Journal of Production Research*, **50**, 5106-5116. http://dx.doi.org/10.1080/00207543.2011.643830

[23] Zhang, X.M. and Han, X.C. (2012) The Balance Problem Solving of the Car Mixed-Model Assembly Line Based on Hybrid Differential Evolution Algorithm. *Applied Mechanics and Materials*, **220-223**, 178-183. http://dx.doi.org/10.4028/www.scientific.net/AMM.220-223.178

[24] Fathi, M., Jahan, A., Ariffin, M.K. and Ismail, N. (2011) A New Heuristic Method Based on CPM in SALBP. *Journal of Industrial Engineering International*, **7**, 1-11.

[25] Fattahi, P., Roshani, A. and Roshani, A. (2011) A Mathematical Model and Ant Colony Algorithm for Multi-Manned Assembly Line Balancing Problem. *International Journal of Advanced Manufacturing Technology*, **53**, 363-378. http://dx.doi.org/10.1007/s00170-010-2832-y

[26] Rahimi-Vahed, A.R., Rabbani, M., Tavakkoli-Moghaddam, R., Torabi, S.A. and Jolai, F. (2007) A Multi-Objective Scatter Search for a Mixed-Model Assembly Line Sequencing Problem. *Advanced Engineering Informatics*, **21**, 85-99.

[27] Chutima, P. and Iammi, J. (2003) Application of Genetic Algorithms in Mixed-Model Assembly Line Balancing. *KMUTT Research & Development Journal*, **26**, 1-16.

[28] Haq, A.N., Jayaprakash, J., and Rengarajan, K. (2006) A Hybrid Genetic Algorithm Approach to Mixed-Model Assembly Line Balancing. *International Journal of Advanced Manufacturing Technology*, **28**, 337-341. http://dx.doi.org/10.1007/s00170-004-2373-3

[29] Su, P. and Lu, Y. (2007) Combining Genetic Algorithm and Simulation for the Mixed-Model Assembly Line Balancing Problem. 3*rd International Conference on Natural Computation* (*ICNC* 2007), **4**, 314-318.

[30] Bai, Y., Zhao, H. and Zhu, L. (2009) Mixed-Model Assembly Line Balancing Using the Hybrid Genetic Algorithm. *International Conference on Measuring Technology and Mechatronics Automation*, **3**, 242-245. http://dx.doi.org/10.1109/icmtma.2009.591

[31] Al-e-Hashem, S.M.J.M. and Aryanezhad, M.B. (2009) An Efficient Method to Solve a Mixed-Model Assembly Line Sequencing Problem Considering a Sub-Line. *World Applied Sciences Journal*, **6**, 168-181.

[32] Moon, I., Logendran, R. and Lee, J. (2009) Integrated Assembly Line Balancing with Resource Restrictions. *International Journal of Production Research*, **47**, 5525-5541. http://dx.doi.org/10.1080/00207540802089876

[33] Akpinar, S. and Bayhan, G.M. (2011) A Hybrid Genetic Algorithm for Mixed-Model Assembly Line Balancing Problem with Parallel Workstations and Zoning Constraints. *Engineering Applications of Artificial Intelligence*, **24**, 449-457. http://dx.doi.org/10.1016/j.engappai.2010.08.006

[34] Mamun, A.A., Khaled, A.A., Ali, S.M. and Chowdhury, M.M. (2012) A Heuristic Approach for Balancing Mixed-Model Assembly Line of Type I Using Genetic Algorithm. *International Journal of Production Research*, **50**, 5106-5116. http://dx.doi.org/10.1080/00207543.2011.643830

[35] Senthilkumar, P. and Shahabudeen, P. (2006) GA Based Heuristic for the Open Shop Scheduling Problem. *International Journal of Advanced Manufacturing Technology*, **30**, 297-301. http://dx.doi.org/10.1007/s00170-005-0057-2

[36] Sivasankaran, P. and Shahabudeen, P. (2014) Study and Analysis of GA-Based Heuristic Applied to Assembly Line Balancing Problem. *Journal of Advanced Manufacturing Systems*, **13**, 113-131. http://dx.doi.org/10.1142/S0219686714500085

[37] Fattahi, P. and Salehi, M. (2009) Sequencing the Mixed-Model Assembly Line to Minimize the Total Utility and Idle Costs with Variable Launching Interval. *International Journal of Advanced Manufacturing Technology*, **45**, 987-998. http://dx.doi.org/10.1007/s00170-009-2020-0

[38] Ozcan, U., Cercioglu, H., Gokcen, H. and Toklu, B. (2010) Balancing and Sequencing of Parallel Mixed-Model Assembly Lines. *International Journal of Production Research*, **48**, 5089-5113. http://dx.doi.org/10.1080/00207540903055735

[39] Yagmahan, B. (2011) Mixed-Model Assembly Line Balancing Using a Multi-Objective Ant Colony Optimization Approach. *Expert Systems with Applications*, **38**, 12453-12461. http://dx.doi.org/10.1016/j.eswa.2011.04.026

[40] Panneerselvam, R. (2012) Production and Operations Management. 3rd Edition, PHI Learning, New Delhi.

[41] Panneerselvam, R. (2016) Design and Analysis of Algorithms. 2nd Edition, PHI Learning, New Delhi.

[42] Panneerselvam, R. (2012) Design and Analysis of Experiments. PHI Learning, New Delhi.

[43] Panneerselvam, R. (2012) Research Methodology. PHI Learning, New Delhi.

[44] Panneerselvam, R. (2006) Operations Research. PHI Learning, New Delhi.