Differential evolution algorithm for solving RALB problem using cost and time based models

J Mukund Nilakantan[1*], Izabela Nielsen[1], S.G. Ponnambalam[2], S. Venkataramanaiah[3]

[1]Department of Mechanical and Manufacturing Engineering, Aalborg University, Denmark
Email: {mnj, izabela}@m-tech.aau.dk

[2]Advanced Engineering Platform and School of Engineering,
Monash University Malaysia, 46150 Bandar Sunway, Malaysia
Email:sgponnambalam@monash.edu

[3]Indian Institute of Management Lucknow, Noida Campus, India
Email: svenkat@iiml.ac.in

Abstract

Assembly process is one of the important aspects in manufacturing industries. Industries are extensively using advanced technologies in assembly lines recently such as robots instead of human labor. Cost associated with human labor such as wages, training, safety and employee management are eliminated with the help of robots. Investments on assembly lines are cost intensive and industries continuously need to maximize their utilization. In this paper, a cost based robotic assembly line balancing problem (RALB) with an objective of minimizing assembly line cost and cycle time is addressed. Moreover, there is no research reported on concurrently optimizing cycle time and assembly line cost for a robotic assembly line system to date. The objective of this paper is to propose models with dual focus on time and cost to minimize the cycle time and total assembly line cost simultaneously. Time based model with the primary focus to optimize cycle time and the cost based model with the primary focus to optimize total assembly line cost is developed. Due to NP-hard nature, differential evolution (DE) is the algorithm used to solve the RALB problem. Straight and U-shaped robotic assembly line problems are solved using the proposed algorithm and the detailed comparison of results obtained are presented. While comparing straight and U-shaped RALB problems, assembly line cost and cycle time obtained by U-shaped RALB problems are better than the straight RALB problems. Proposed models have significant managerial implications and these have been discussed in detail.

Keywords: Robotic Assembly Line Balancing, Assembly Line Cost, Cycle Time, Differential Evolution.

1. Introduction

In a manufacturing sector, assembly process is considered to be one of the most critical tasks. Assembly lines are developed for cost-efficient mass production to make full use of labor and resources available [1]. Stiff competitive environment requires the industries using assembly lines to produce products at a very low cost without comprising on the quality of the product in a reasonable time. To remain competitive, the manufacturers need to speed up the time to market

and at the same time to minimize the manufacturing cost [2]. Different set of tasks are to be executed in a set of predefined workstations in a time efficient manner in an assembly line. Assembly lines are to be designed in such a way that tasks are grouped to workstations in an orderly manner so that line efficiency is maximized and this problem of dividing the tasks to the workstation in a balanced manner is classified as assembly line balancing (ALB) problem [3]. Cost-oriented assembly line balancing is a generalized form of time-based assembly line balancing [4].The major objective in a cost-oriented assembly line balancing problem is to assign all tasks to the workstations in such a way that precedence relationship are met and the production cost is minimized [5]. Both short and longer term operating costs are incorporated for solving the cost based line balancing problems. Labor costs, setup cost, equipment cost and inventory cost have been considered to solve this type of problems [6].

Different researchers have applied exact methods, heuristics and metaheuristics to solve cost based assembly line balancing problems. Two new heuristics Wage Rate Method (WR) and the Wage Rate Smoothing-Method (WRS) developed by Rosenberg and Ziegler [5] focused on solving the cost based assembly line balancing problem with an objective of minimizing the total production cost. The experimental results obtained are compared with the well-known heuristics: Positional Weight Method (PW) [7] and the Positional Weight Wage Rate Difference Method (PWWD) [8]. From the results reported it is concluded that PWWD and WRS are superior to PW and WR. Amen [4] proposed a cost based assembly line balancing model for a single model assembly line with the objective of minimizing the total cost per unit. For solving the problem, an exact backtracking technique is used. The experimental results show that the proposed method finds optimal solution for small (50 tasks) and medium sized (75 and 100 tasks) problems in reasonable computational time. Amen [9] considered scenarios where production is very labor-intensive and wage rates are based on the requirements and capabilities of the workforce. Two new heuristics were developed to solve this problem. Comparison on the quality of the solution and computational time of the developed algorithm are reported in [10]. Amen's study is used as the basis of the research work of Scholl and Becker [11] and it is shown in their work that one of the rules developed by Amen is incorrect and presented a corrected and simplified version of this rule. Padrón et al. [2] presented a line balancing methodology which combines a heuristic model and exact algorithm with an objective of minimizing cost in a feasible computational time. Cost function considered includes short term

operating costs, task and work station capital investment costs. Erel et al. [12] proposed a beam search algorithm which is similar to Tabu Search (TS) to solve an assembly line balancing problem in U-shaped assembly line with an objective of minimizing total labor cost and total expected incompletion cost. The performance of the proposed algorithm is compared with the other algorithms reported in the literature and it is analyzed that the proposed algorithm performs better. Roshani et al. [13] developed a simulated annealing algorithm for a cost based two-sided assembly line balancing problem. Proposed algorithm is tested on different problems to test the effectiveness of the algorithm. The literature review reveals that the literatures on cost based assembly line balancing problems are relatively scarce. However, Hazır et al. [14] presented a survey paper in which problems, approaches and analytical models on cost based assembly line balancing are analyzed in detail.

Robotic assembly line balancing (RALB) problems is an extension of simple assembly line balancing (SALB) problems [15]. Robotic assembly line balancing (RALB) problem aims at assigning tasks to the workstation and selecting the robot to perform the allocated tasks for each workstation in an efficient manner such that the productivity is improved. Technological advancements help in replacing the human labor with robots which can perform all types of tasks in an assembly line. Robots help in improving the productivity, flexibility and provide a safe environment for the labor. Different types of robots are available in the market and are extensively used in assembly lines recently. An example of a typical robotic assembly line in a shop floor is presented in Figure 1. Workstations in this assembly line are arranged in a straight line and different types of robots are allocated to these workstations to perform the tasks in the workstations. Robotic assembly line works in a collaborative manner with other resources (e.g., automated guided vehicle and human labor) in the shop floor such for a smooth assembly operation. In a robotic assembly line, selection of the best performing robot to complete the tasks in a workstation is a very critical issue [16].Quality of the assembly line depends on the robot assignment. Researchers have so far focused on objectives such as minimizing cycle time [17], minimizing number of workstations, maximizing line efficiency [18] and minimizing energy consumption of the robots [19].

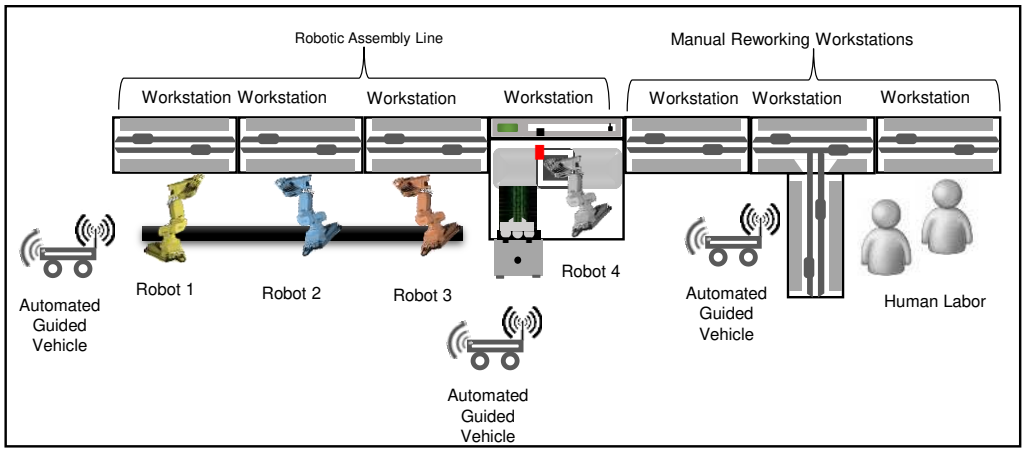


Figure 1 Robotic Assembly Line System

Researchers have classified simple assembly line balancing (SALB) problems in the category of NP-hard and the proposed problem in this paper is an extension of SALB problems and falls in this category. Detailed literature on different optimization techniques (exact methods, heuristics and metaheuristics) to solve assembly line balancing problems are reported in [20]. Table 1 presents a summary of relevant literature review of different works related to cost and time based assembly line balancing problems in both traditional and robotic assembly lines where different optimization techniques have been used.

From the table it could be seen that researchers have focused on assembly line balancing problems with focus on the objective of minimizing time and cost. It could also be analyzed that researchers focused on the objective of minimizing cycle time in the robotic assembly lines and no work has been reported on the objective of minimizing production cost in a robotic assembly line. There is a need to propose models for robotic assembly line balancing problems with the objective of minimizing production cost and cycle time as these types of assembly lines are widely used in a number of industries and optimizing this objective is a very critical.

The main contributions of this paper are: 1) Two models for robotic assembly line balancing problem are proposed. First model focuses on minimizing the total production cost of a robotic assembly line and cycle time is evaluated. Second model focuses on minimizing the cycle time of a robotic assembly line and the total production cost incurred is calculated. Proposed models are evaluated for two types of robotic assembly line (Straight and U-shaped). 2) A mathematical model for the proposed problem is presented. 3) Differential evolution is a metaheuristic developed to solve the proposed problem due to its NP-hard nature. Remainder of the paper is structured as follows. Section 2 explains the problem in detail and presents the mathematical

model. Section 3 presents the details of differential evolution, metaheuristic algorithm used to solve the proposed RALB problem. Section 4 reports the detailed experimental results conducted. Section 5 concludes the findings of this work and the managerial implications of the proposed work.

Table 1 Summary of literature on assembly line balancing problems

Reference	Assembly line configuration	Type of Objective		Objective	Methodology used to solve the problem	Remarks including real life problem or hypothetical
		Cost based	Time based			
Rosenberg and Ziegler [5]	Traditional straight assembly line	√	--	Minimizing the total production cost	Heuristics	Randomly generated problems
Amen [2]	Traditional straight assembly line	√	--	Minimizing the total cost per unit	Exact backtracking technique	Randomly generated problems
Amen [5]	Traditional straight assembly line	√	--	Minimizing the total cost per unit	Heuristics	Randomly generated problems
Padrón et al. [2]	Traditional straight assembly line	√	--	Minimizing short term operating and work station investment costs	Heuristic and Exact algorithm	Benchmark ALB problems
Erel et al. [8]	Traditional U-shaped assembly line	√	--	Minimizing total labor cost and total expected incompletion cost	Beam search algorithm	Benchmark ALB problems
Roshani et al. [9]	Two sided assembly line	√	--	Minimizes the total cost per product unit	Simulated annealing algorithm	Benchmark ALB problems
Levitin et al. [15]	Straight robotic assembly line	--	√	Minimize the cycle time	Genetic algorithm	Randomly generated problems
Gao et al. [16]	Straight robotic assembly line	--	√	Minimize the cycle time	Hybrid Genetic algorithm	Benchmark problems
Nilakantan et al. [17]	Straight robotic assembly line	--	√	Minimize the cycle time	Particle swarm optimization	Benchmark problems
Yoosefelahi et al. [18]	Straight robotic assembly line	√	√	Minimize the cycle time, robot costs and setup cost	Multi objective evolution strategies	Benchmark problems
Nilakantan et al. [19]	Straight robotic assembly line	--	√	Minimize cycle time and energy consumption	Particle swarm optimization	Benchmark and Randomly generated problems
Mukund Nilakantan and Ponnambalam [21]	U-shaped robotic assembly line	--	√	Minimize the cycle time	Particle swarm optimization	Benchmark problems

2. Problem definition and Mathematical Model

In a robotic assembly line, at each workstation different assembly tasks are performed to assemble a product. Precedence constraints of the tasks are predefined and it determines the order in which tasks should be executed. In a robotic assembly line there will be a set of workstations and robots. In a balanced assembly line, tasks are allocated to the workstations and best available robot to perform the allocated tasks is to be chosen. The main objectives considered in this paper are to assign tasks to the workstations and assign the robots which will perform the tasks with minimum cost (cost based model) and minimum cycle time (time based model) when the number of workstations is fixed. The following assumptions considered in the model formulation are similar to those mentioned in [15] and [16].

The assumptions considered for the RALB problem are the following.

1. Robot initial cost includes installation, maintenance and service cost for the entire service life. The service life is restricted to five years. The robot initial costs are assumed based on the literature.
2. Robots are assumed to work for 20 hours a day and 300 days in a year.
3. Using annual fixed interest rate of 10%, equivalent uniform annual costs of all the robots are calculated.
4. There is no limitation in the availability of the robots. In this paper, number of robots is considered to be the same as the number of workstations.
5. Problem is designed for a straight and U-shaped assembly line system where a unique model of a single product is to be assembled.
6. Tasks cannot be subdivided and it should meet precedence constraints.
7. All robots are available without any limitations (i.e., number of robots of same capability is unrestricted).
8. Time taken to perform a task depends on the robot assigned. Material handling, loading and unloading times of the components in the assembly line, as well as set-up and tool changing times are negligible, or are included in the activity times. This assumption is realistic for a single model assembly line, where a single product is assembled. In such robotic lines, tooling is designed such that tool changes are minimized. The performance time of the robots utilized in this paper are adopted from the datasets reported by Gao et al. [16].

A zero-one integer programming (IP) model for this problem when the objective is to minimize the total production cost is formulated in this section. The cost based model for straight robotic assembly line is presented. The following notations are used in this paper:

Indices and Parameters

i, j: Index of assembly tasks

s: index of work stations, $s= 1, 2... N_w$

h: index of robots, $h= 1, 2... N_r$

N_w: Number of workstations

N_a: Number of tasks

N_r: Number of robots

C: Cycle time

c_{ih}: cost of performing the task *i* by robot *h*

t_{hi}: processing time of task *i* by robot *h*

pre(i): set of immediate predecessors of task *i*

Decision Variables

$$x_{is} = \begin{cases} 1 \text{ if task } i \text{ is assigned to workstation } s \\ 0, \text{ otherwise} \end{cases}$$

$$y_{sh} = \begin{cases} 1 \text{ if robot } h \text{ is allocated to workstation } s \\ 0, \text{ otherwise} \end{cases}$$

Model Formulation:

$$\text{Min } Cost = \sum_{i=1}^{N_w} \left\{ \sum_{i=1}^{N_a} \sum_{i=1}^{N_w} c_{ih}.x_{is}.y_{sh} \right\} \quad (1)$$

Subject to:

$$\sum_{s=1}^{Nw} s.x_{is} - \sum_{s=1}^{Nw} s.x_{js} \leq ,0 \quad \forall i \in pre(j); j \quad (2)$$

$$\sum_{s=1}^{N_W} x_{is} = 1 \qquad \forall i \tag{3}$$

$$\sum_{s=1}^{N_W} y_{sh} = 1 \qquad \forall s \tag{4}$$

$$x_{is} \in \{0,1\} \quad \forall s,i \tag{5}$$

$$y_{sh} \in \{0,1\} \quad \forall h,s \tag{6}$$

The objective of the cost based model (Equation 1) is to minimize the total assembly line cost. Equation 2 defines the precedence relationship among the tasks. It ensures that for a pair of tasks with precedence relation, the precedent task cannot be assigned to a workstation after the one to which its successor is assigned. Equation 3 ensures that each task has to be assigned to one workstation and Equation 4 ensures that each workstation is equipped with one robot. It is notable that objective function is non-linear. Hence, it is hard for traditional exact optimization techniques to solve the problem.

Mathematical model for U-shaped cost based robotic assembly line is presented below. For a given set of tasks F= {$g \mid g$ = 1, 2, .. .,n}, a set of precedence constraints P = { (i, j) | task i must be completed before task j }, a set of task times T = { $t(g) \mid g$ = 1, 2, ..., n }, and a cycle time C, find a collection of subsets of F, (L_1, L_2, . . . , L_N) where L_a = {g| task g is done at workstation a } and the workstations and tasks are arranged in a U-shape. In case of U-shaped robotic assembly line precedence relationship equation changes and hence Equation 2 is replaced by Equation 7.

For each task j:

$$\begin{aligned} &\text{if } (i,j) \in P, i \in L_a, j \in L_b, \text{then } a \le b, \text{ for all } i; \text{or} \\ &\text{if } (j,k) \in P, y \in L_b, k \in L_c, \text{then } c \le b, \text{ for all } k; \end{aligned} \tag{7}$$

The mathematical model for robotic assembly line balancing problem with the objective of minimizing cycle time in straight and U-shaped robotic assembly line are presented in [17] and [21].

3. Metaheuristic algorithm to solve RALB problem

Assembly line balancing problems falls under the category of NP-hard and many researchers have proposed metaheuristic algorithms to solve different types of these problems [22]. Detailed literature on different metaheuristic algorithms used to solve assembly line balancing problems are presented in [23]. The problem addressed in this paper is also NP-hard and to solve the problem, differential evolution algorithm is proposed. Detailed description on how the metaheuristic algorithm is implemented is presented in this section.

3.1 Differential Evolution

Differential evolution (DE) is an metaheuristic algorithm proposed by Storn and Price [24] for solving optimization and engineering problems. Due to its simplicity in implementation, DE has been applied to solve real-world problems like job shop scheduling and engineering design optimization [25]. DE has three parameters which controls the search process. Process of selecting the parameters is explained later in the paper. DE is very similar to genetic algorithm; however main differences are in the mechanism of mutation and crossover operation [26]. DE has been chosen for solving this problem mainly due to the following advantages [27]: a) able to find the true global minimum regardless of the initial parameter values, b) fast convergence, and c) few control parameters to fine tune. A random set of initial population composed of target vectors are generated initially. This population undergoes the evolution process in a form of natural selection. Mutation, crossover and selection operators are applied for generating new population with higher quality. Each target vector undergoes the mutation operation to generate a set of donor vectors for all iterations. A set of trial vectors is created by undergoing a crossover operation on target and donor vector. The selection operation is performed by comparing the fitness values of each target vectors and trial vectors. If the fitness value of trial vector is better than the fitness value of the target vector, then trial vector will be selected into the population otherwise target vector will be selected. The above mentioned three processes are repeated until the termination condition is satisfied.

3.2 Research Design (selection of DE algorithm parameters)

DE algorithm utilizes different parameters and in this paper different inputs and parameter values are selected based on literature. Based on the preliminary experiments conducted for the problem

under study, different parameters selected are shown in Table 2. This section also provides the details of different components of DE.

Table 2 DE Algorithm parameters

Parameter	Value	Reference(s)
Initial Population	25	[28]
Mutation factor	0.5	[29]
Crossover	Ordered Crossover; Crossover rate: 0.9	[30]
Selection	Based on the objective function (minimizing assembly line cost and cycle time)	--

A) Population Initialization

The main step in the functioning of the DE is the generation of the initial population. Each member (vectors) of this population encodes a potential solution for the problem. Vector represents a sequence of numbers (tasks) arranged in such a way that it meets the precedence relationship. Instead of starting the algorithm with a random population, a set of priority dispatching rules reported in the literature are used to generate the set of initial population. Six rules reported in the literature[28] are selected to create the initial population and remaining vectors are randomly generated. Detailed explanation on how the vectors are generated for robotic assembly line balancing problems are presented in [19].Each vector in the population is evaluated for the objective function (fitness value). Section 3.3 presents the detail of the procedure followed to evaluate the objective function.

B) Mutation

In DE, mutation is one of the prime operations. Mutation process is performed for all the vectors in the population. Mutation process at each generation is performed by picking three target vectors from the population. Using the target vectors, a population of donor vectors is created. Perturbation is performed by adding the difference between the two randomly picked target vectors to a third target vector. This is done based on the Equation 8.

$$y_{ig} = x_{r1,g} + M(x_{r2,G} - x_{r3,G}), where\ i\ = 1,...5 \qquad (8)$$

M is known as the mutation scaling factor.

To show the process of mutation in RALB problem an example is illustrated below.

Let the three vectors be:

$x_{r1,G}$={1,2,6,3,4,5,7,8,10,9,11} $x_{r2,G}$={1,2,3,4,5,6,7,8,9,10,11}and

$x_{r3,G}$={1,2,3,6,5,4,7,8,10,9,11}, M=0.5

y_{ig}={1,2,6,3,4,5,7,8,10,19,11}+0.5*{1,2,3,4,5,6,7,8,9,10,11}-{1,2,3,6,5,4,7,8,10,9,11}

The pairs of transpositions to get $x_{r3,G}$ from $x_{r2,G}$are identified. Mutation factor is applied to select the number of pairs and these selected pairs is used to transposition the values in $x_{r1,g}$.

y_{ig}={1,2,6,3,4,5,7,8,10,19,11}+0.5*(3,5)(8,9)={1,2,6,3,4,5,7,8,10,19,11}+(8,9)={1,2,6,3,4,5,7,8,9,10,11}

C) Crossover

Crossover operations are performed after the mutation operation is completed. By choosing a donor vector and target vector a set of trial vectors are generated. Crossover operation is performed only for a selected set of vectors in the population. Using a crossover rate C_R, number of vectors for crossover is selected. OX operator (order crossover) proposed by Davis [30] is adopted in this research to generate trial vectors.

The detailed description of the OX operation is explained below.

- A subsection of the task sequence from the target vector is picked randomly.
- A proto-trial vector is created by copying the substring of the task sequence into the corresponding positions.
- Remove redundant tasks in the substring from the donor vector. Formed sequence of tasks contains the tasks that the proto-trial vector needs.
- Place the tasks into the unfixed positions of the proto-trial vector from left to right according to the order of the sequence in the donor vector.

To explain this method an example is shown in Figure 2.

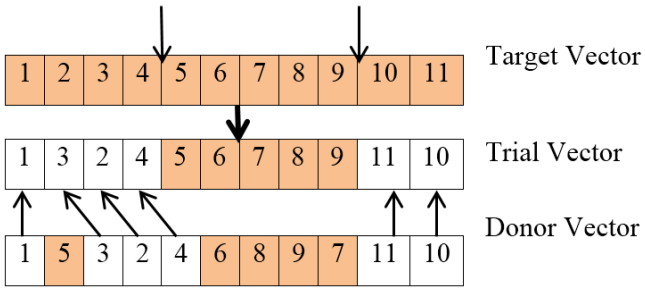


Figure 2 Illustration of the OX operator

A reordering procedure used by (Levitin et al., 2006) is also incorporated to make the vectors feasible if the created vector does not meet the precedence constraints

D) Selection

Selection procedure is different from other metaheuristic algorithms. Population of the next generation is selected from the individual in the current population and its corresponding trial vector. The vector with the better fitness value is copied to the next generation. Based on the objective of minimizing the total production cost and cycle time, the selection operation picks the vectors for the next iteration. Rule of selection is based on the following rule.

$$x_{i,\mathrm{G}+1} = \begin{cases} y_{i,G} & \text{if } f(y_{i,G}) < f(x_{i,G}) \\ x_{i,G}, & \text{otherwise} \end{cases} \qquad (9)$$

The differential evolution algorithm is terminated if the iteration approaches a predefined criteria, usually a sufficiently good fitness or in this case, a predefined maximum number of iterations (generations) is used.

3.3 Fitness value evaluation

The fitness value to be evaluated in this research is to minimize the total production cost (cost based model) as well as minimize the cycle time (time based model) of the robotic assembly line. In the cost based model, the allocation of tasks and robots are performed by minimizing the total production cost. The subsequent cycle time of that allocation is also evaluated. In case of time based model, the allocation of tasks and robots are performed with an objective of minimizing cycle time. The subsequent production cost of the allocation is also evaluated. In this paper, both straight and U-shaped robotic assembly lines are presented.

3.3.1 Cost and time based model- straight robotic assembly line

Consecutive allocation procedure is adopted for task and robot allocation with an objective of minimizing the total assembly line cost (cost based model). This allocation procedure aims at assigning tasks to the workstation and allocates the best robot which performs these tasks with a minimum performance cost. An initial assembly line cost is to be calculated to start the procedure. The initial assembly line cost is determined using Equation 10. The procedure tries to allocate the maximum tasks to each workstation for the initial assembly line cost. If the procedure cannot find the optimal allocation within the initial value, the initial value is incremented by one and the procedure is repeated until all the tasks get assigned to the given number of workstations.

Initial assembly line cost $P_0 = \left\lceil \sum_{j=1}^{N_a} \min_{1 \le i \le N_r} c_{i,j} / N_w \right\rceil$ (10)

An example is used to explain the procedure involved in consecutive allocation procedure.

- Example task sequence (Generated based on priority rules as explained in Section 3.2): ***(1-4-5-3-7-9-2-6-8-10-11)***
- Total number of robots and workstation is 4.

Step 1. Minimum cost to perform each task by any robot among the given set of robot is used to calculate the initial assembly line cost P_0. In the given example below initial P_0 is found out to be 98 (refer Table 3).

$$P_0 = [33+40+35+36+24+57+37+31+31+36+33]/4=98.$$

Step 2. Procedure tries to allocate the first task to the first workstation and check if any of the robots can perform the tasks within the initial assembly line cost.

Step 3. If yes, next immediate task in the sequence is checked if it can be allotted to the same workstation within the initial assembly line cost.

Step 4.The procedure is repeated until the workstation is able to handle the tasks allotted within the initial P_0 value.

Step 5.If the first workstation cannot accommodate further tasks, the next workstation is opened and tasks are allotted.

Step 6. Repeat this procedure until all the tasks are allotted and robots are assigned.

Step 7. For the initial P_0, if there are tasks still left unassigned, P_0 is incremented by 1 and the procedure is repeated until all tasks gets allotted.

Step 8. Best robot which can perform the allotted tasks is selected based on the minimum performance cost.

Step 9. The overall assembly line cost is calculated by summing up the cost of performing the allotted task in each workstation by the allocated robots.

Using the performance cost and precedence relations data presented in Table 3, the given sample sequence is evaluated. Cost data is generated randomly and details of the method followed for dataset generation is presented in Section 4. Time of performing tasks by different robots are available in [16].Allocation of tasks when P_0 is 98 is shown in Figure 3 and it is observed that tasks 9, 10, and 11 are left unassigned. To allocate all the tasks P_0 is incremented till 137 for the complete allocation as shown in Figure 4. Cost of each workstation is calculated

and total assembly line cost is calculated by summing the cost to perform the tasks at each workstation. For the given sequence of tasks, the total assembly line cost is calculated as 429.For a sample sequence **(1-2-3-4-5-6-7-8-9-10-11)**,the allocation of tasks are done based on the cost based model for a straight robotic assembly line with the objective of minimizing the total assembly line cost. The cycle time of the allocated tasks of the straight robotic assembly is evaluated using the time data for the problem. The time to perform the tasks in each workstation allocated based on the cost model is calculated based on the task performance data. Table 4 shows the task and robot allocation for the sample sequence for cost based model. Figure 5a) shows the workstation times and assembly line cost of each workstation calculated based on cost based model. The workstation time is calculated using the time data available in Table 3. Time at Workstation 1 (Robot 4) =49+42+52=143, Time at Workstation 2 (Robot 2) = 41+36+65=142, Time at Workstation 3 (Robot3) =40+34+41=115 and Time at Workstation 4 (Robot 2) = 46+38=84. The cycle time (C.T.) is 143 and the total assembly line cost is 441

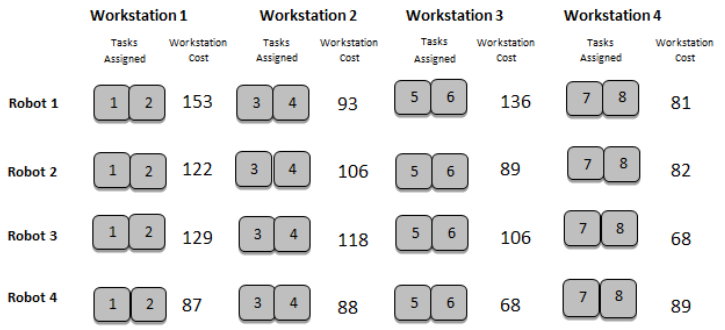


Figure 3 Allocation done for initial assembly line cost

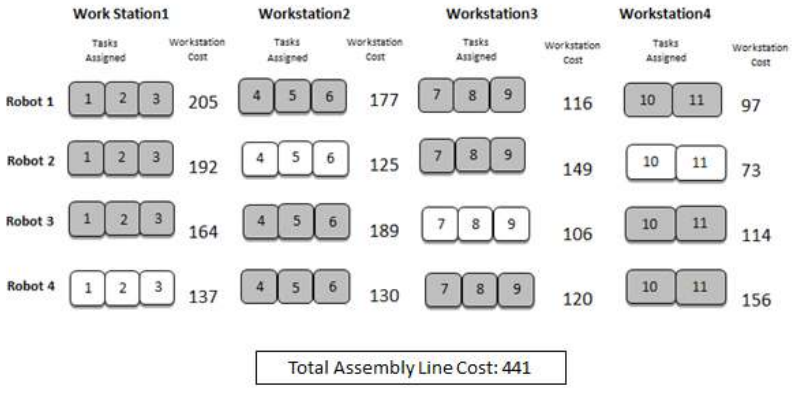


Figure 4 Final allocation of tasks and robots using cost based model for a straight RAL

To illustrate the time based model, the following tasks sequence **(1-3-2-4-5-6-7-9-8-10-11)** is used. The objective of the time based model is to allocate the tasks to the workstations with an objective of minimizing the cycle time. In this research, time based model is similar to the one presented in [19] and cycle time is evaluated. The total assembly line cost of the allocation done based on this model is also evaluated. Based on this allocation, using cost of the performing the tasks presented in Table 3, the cost of the assembly at each workstation is calculated. And the overall assembly line cost is calculated by taking the sum of the cost to perform the tasks at each workstation. Cost for Workstation 1 (Robot 4) =47+40+50= 137, Cost for Workstation 2 (Robot 4) = 38+24+68= 130, Cost for Workstation 3 (Robot 3) =37+31+38= 106 and Cost for Workstation 4 (Robot 2) = 40+33=73. Table 5 shows the allocation of tasks and robots allotted using the time based model and their subsequent costs and workstations times. Figure 5b) shows the robot and task allocation with the workstation cost and workstation time in a straight robotic assembly line calculated based on time based model. The cycle time is 143 and the total assembly line cost is 446.

When comparing Figure 5a) and Figure 5b), one can find that the allocation of tasks in both the models are same; however there is a difference in allocation of robots to the workstations which results in different cycle time and workstation cost for the models. This is due to the difference in the objective functions of each model.

Table 3 Input data for 11 task and 4 robot problem

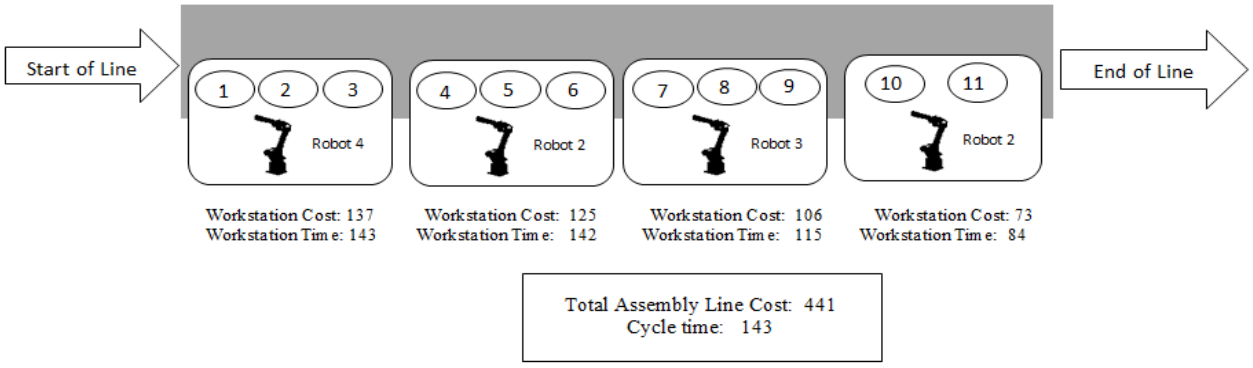
Task	Precedence Relations	Cost for performing the tasks				Time for performing the tasks			
		R1	R2	R3	R4	R1	R2	R3	R4
1	-	65	33	47	47	81	37	51	49
2	1	88	89	82	40	109	101	90	42
3	1	52	70	35	50	65	80	38	52
4	1	41	36	83	38	51	41	91	40
5	1	74	32	30	24	92	36	33	25
6	2	62	57	76	68	77	65	83	71
7	3,4,5	41	45	37	47	51	51	40	49
8	6	40	37	31	42	50	42	34	44
9	7	35	67	38	31	43	76	41	33
10	8	36	40	38	73	45	46	41	77
11	9,10	65	33	47	47	76	38	83	87

Table 4 Task and robot allocation using cost based model

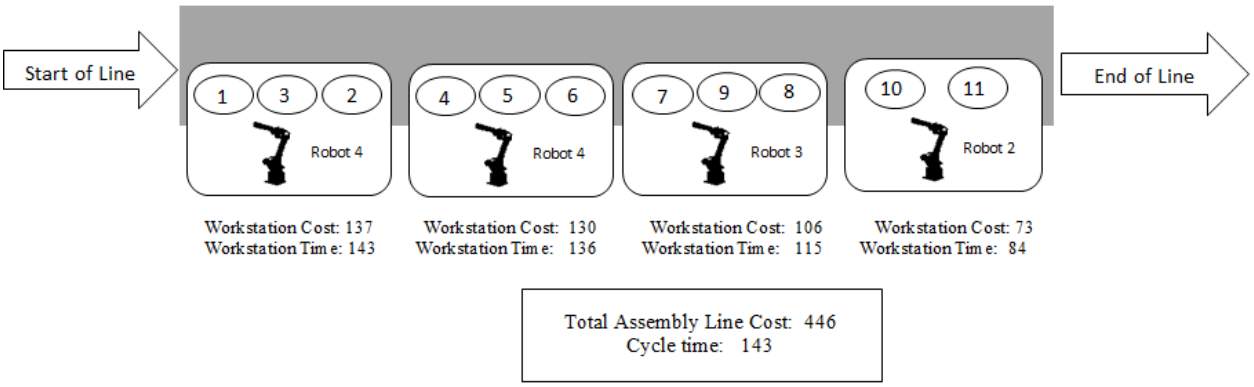
Workstation	Tasks	Robot Allotted	Workstation Cost	Workstation Time
Workstation 1	1, 2, 3	Robot 4	137	**143**
Workstation 2	4, 5, 6	Robot 2	125	142
Workstation 3	7, 8, 9	Robot 3	106	115
Workstation 4	10, 11	Robot 2	73	84
Total Assembly Line Cost			**441**	**C.T.-143**

Table 5 Task and robot allocation using time based model

Workstation	Tasks	Robot Allotted	Workstation Cost	Workstation Time
Workstation 1	1, 3, 2	Robot 4	137	**143**
Workstation 2	4, 5, 6	Robot 4	130	136
Workstation 3	7, 9, 8	Robot 3	106	115
Workstation 4	10, 11	Robot 2	73	84
Total Assembly Line Cost			**446**	**C.T.-143**



a) Cost based model allocation



b) Time based model allocation

Figure 5 Workstation cost and cycle time for straight RAL

3.3.2 Cost and time based model- U-shaped robotic assembly line

This section presents the detailed procedure implemented to calculate the total assembly line cost of a U-shaped robotic assembly line. U-shaped robotic assembly line allows more possibilities of task allocation when compared with straight robotic assembly line. Allocation of tasks to the workstation are done by moving forward and backward based on the precedence relation in contrast to the typical forward move in a straight robotic assembly line. An initial assembly line cost (P_0) is calculated to start the procedure. The procedure tries to allocate the maximum number of tasks to the workstations without violating the precedence constraints. If the initial P_0 cannot accommodate all the tasks, $P_0$is incremented by one and the procedure is repeated to accommodate all the tasks. Based on the allocation done, cost of performing the tasks allotted to the workstation by a robot which can perform the allocated task with minimum cost is calculated. The total assembly line cost is calculated by taking the sum of cost incurred at each workstation. An illustration is provided in this section which explains the task and robot allocation and calculation of total assembly line cost in a U-shaped robotic assembly line. Sequence of tasks which meets the precedence constraints is considered for illustration. Let the sequence of tasks be, (**1-2-3-4-5-6-7-8-10-9-11**): 11 task and 4 workstation problem is considered for the illustration. Performance cost data details of each tasks and robots are presented in Table 3.

Step 1. Using Equation 10, P_0 is calculated and it is found to be 98.

Step 2. For the initial P_0, the procedure tries to allocate the tasks to the workstations starting from the first workstation. Procedure checks the both sides of the sequence if any of the robots could perform the tasks within P_0. Due to the characteristic of U-shaped assembly line, different possible task combinations are available. This procedure chooses the task combination which minimizes the cost at each workstation.

Step 3.If the initial assembly line cost cannot accommodate all the tasks, next workstation is open and remaining tasks from the sequence are allocated.

Step 4. The initial value of assembly line cost is incremented by one if tasks are still left unassigned for the initial value and Step 2 and 3 are repeated until all tasks get assigned to the workstation.

Step 5.Based on the allocated tasks, the robots which can perform these allocated tasks are chosen based on the minimum cost.

Step 6. The sum of cost of each workstation gives the total assembly line cost of the given task sequence.

In the given example, when the allocation is attempted with initial P_0 it is found that tasks 5, 6, 7 and 10 are left unassigned. Hence, P_0 is incremented till 125 to accommodate all the tasks to the four workstations. The total assembly line cost of the given sequence is calculated as 416 cost units. Figure 6 shows the allocation based on the cost based model in a U-shaped RAL. Based on the allocation done using cost based model, the cycle time of the allocation is calculated using the task performance times shown in Table 3. Time at Workstation 1 (Robot 2) =37+46+38=121, Time at Workstation 2 (Robot 4) = 22+33+44=119, Time at Workstation 3 (Robot 3) =38+40=78 and Time at Workstation 4 (Robot 2) = 41+36+65=142. The cycle time of the U-shaped robotic assembly line is 142 and the total assembly line cost is 416 as shown in Figure 7a).

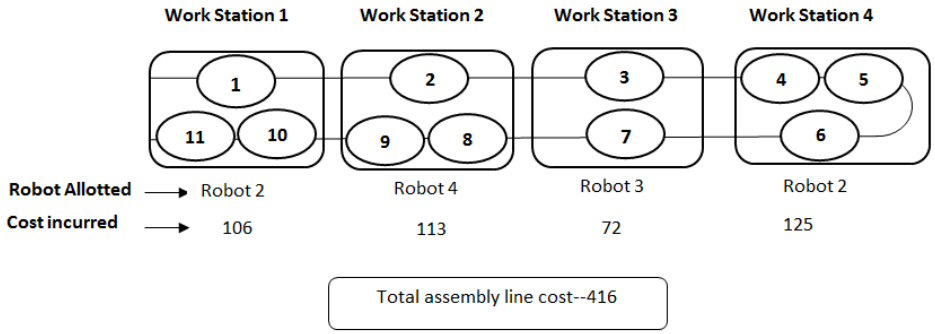
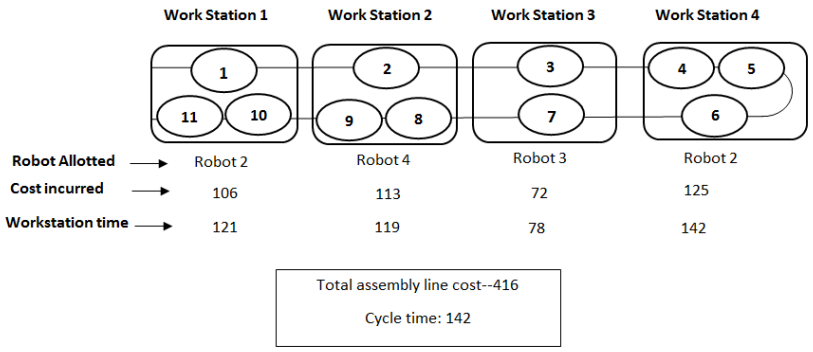


Figure 6 Final task and robot allocation in a U-shaped RAL for cost based model

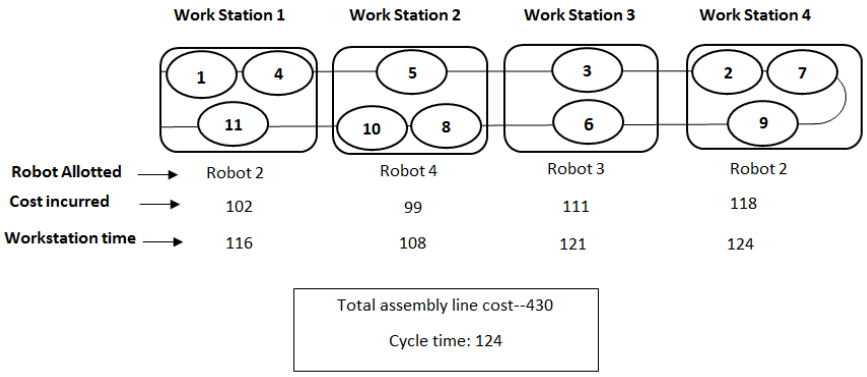
Using the time based model for U-shaped robotic assembly as shown in [21], where the objective is to minimize the cycle time is adopted in this research. Based on the allocation done based on the time based model, the subsequent the total assembly line cost of the U-shaped robotic assembly line is evaluated. Figure 7b) shows the final allocation of tasks and robots based on the objective of minimizing the cycle time (time based model) and using Table3, the overall assembly line cost is calculated by taking the sum of the cost to perform the tasks at each workstation. Cost for Workstation 1(Robot 2) =33+36+33=102, Cost for Workstation 2(Robot 3) = 30+31+38=99, Cost for Workstation 3(Robot 3) =35+76=111 and Cost for Workstation

4(Robot 4) = 40+47+31=118. The cycle time is 124 and the total assembly line cost is 430 when the allocation is done based on the objective of minimizing the cycle time (time based model) in a U-shaped robotic assembly line.

When comparing Figure 7a) and Figure 7b), one can find the difference in the total assembly line cost and the cycle time when allocation are done based on the two models. It can be seen that cost based model is able to find a possible allocation with lower assembly line cost and lower cycle time when compared to time based model.



a) Cost based model allocation



b) Time based model allocation

Figure 7 Workstation cost and cycle time for U-shaped RAL

3.3.3 Evaluation of the models and configurations

This section presents a comparison of solutions obtained using two models for straight RALB and U-shaped RALB. When comparing the cost obtained for straight RALB and U-shaped, solutions obtained using the cost based model is better than the solutions obtained using the time

based model and while comparing the cycle time, solutions obtained by time based model is better than the solutions obtained using cost based model.

While comparing the cycle time and cost of U-shaped RALB with strraight RALB, it can be seen that U-shaped RALB is having lower cycle time and cost. This is due to the different possible (forward and backward) allocations allowed in U-shaped RALB, whereas straight RALB allows only one way of allocating (forward) the tasks. Table 6 presents the comparison of the solutions obtained for 11 tasks - 4 robot problem for both straight and U-shaped RALB using cost and time based model. Percentage improvement of using U-shaped configuration over straight line is also presented and for the problem illustrated it can be concluded that U-shaped is performing better than the straight line interms of assembly line cost and cycle time.

Table 6 Comparison of models and layout of RALB

	Straight RALB		U Shaped RALB		% Improvement (of U-shaped over straight line)
Cost based model	**Assembly Line Cost**	441	**Assembly Line Cost**	416	6.01
	Cycle Time	143	**Cycle Time**	142	0.70
	Cost Improvement %	1.12	**Cost Improvement %**	3.26	--
Time based model	**Assembly Line Cost**	446	**Assembly Line Cost**	430	3.72
	Cycle Time	143	**Cycle Time**	124	15.32
	Cost Improvement %	0.00	**Cost Improvement %**	14.52	--

4. Experimentation and Discussion of Results

To demonstrate the effectiveness of the proposed algorithms for straight and U-shaped robotic assembly line, computational experiments are conducted. The following section describes the experiments conducted.

4.1 Dataset for computational experiments

There are no cost data available to optimize the assembly line cost for a robotic assembly line. This section presents the procedure followed to generate the cost data for the RALB problem. Eight representative precedence graphs and from http://www.assembly-line-balancing.de/, which are widely used in the SALB-I literature [31]and processing times of robots available in [16]are used to generate the datasets. The hourly rate of the robots is calculated from the standard procedure of finding annual cost of a capital intensive resource.

$$UAC = IC * (A/P, i, n) \tag{11}$$

Here, *UAC* = Equivalent uniform annual cost ($/yr); *i* = annual interest rate and *n*= number of years, (*A/P, i, n*) = capital recovery factor that converts initial cost at year 0 into a series of equivalent uniform annual yea-end values.For given values of *i* and *n*, (*A/ P, i, n*) can be computed as follows:

$$(A/P,i,n)=\frac{i*(i+1)^n}{(i+1)^n-1} \qquad (12)$$

Value of (*A/ P, i, n*) can also be found in interest tables that are widely available.

- Hourly cost of robot is calculated by dividing the annual cost with total annual hours per year. Cost of robot for a specific time can be calculated with hourly cost of robot.
- The annual interest rate *i* is assumed as 10% and *n* is assumed as 5 years.
- Number of annual hours per year is calculated as total working hours multiplied by total number of working days. Number of annual hours is taken as 6000hr/yr (20hr/day*300days/yr).
- After calculating the cost per hour of a robot, cost of performing a set of task by a robot is calculated by using the performance time.

An example is shown for a better understanding on how the cost data is generated. The steps shows how the cost of a robot for a specific time. Initial robot cost is $1,100,000.

Step 1: Calculate UAC for robot

UAC = *IC (A/P, i, n)*

= 1,100,000* 0.2638), Uniform Annual Cost = $ 29, 0180

**A/P* Value is calculated for 5 years with interest rate 10%

Step 2: Calculate Hourly Rate of the robot

Total number of hours per year = (20 hr/day) (300 day/yr) = 6000 hr/yr.

Cost Per Hour = 290180/6000

= $ 48.36333/hr

**Assembly line is considered to work for 20 hours a day for 300 days in a year.*

Step 3: Cost of the robot for a specific time

Time taken to perform a task 1 by robot 1 is 81minutes.

Cost of robot per time = 48.3633*81 /60 = $ 65.2905

32 problems are generated using the above mentioned rules. It is assumed that costs such as robot cost, setup cost, transportation cost are included in the initial cost of the robot. Table 3 is developed based on the *UAC* cost and subsequent tasks times of robots available. Appendix B shows the random robot cost generated for developing datasets for small size datasets (up to 70 tasks problems) and Appendix Cfor large size datasets (above 89 tasks problems).

4.2 Parameter Selection for DE

Performance of DE mainly relies on the parameters selected. Parameters are selected based on the tests conducted in order to get a satisfactory solution quality in an acceptable time span. Influence of each parameter on the solution quality is tested. Three datasets of different task size are chosen to find the best combination of parameters. Following are the parameters tested and used in DE to solve the proposed RALB problem:

Stopping Conditions: The proposed DE algorithm is terminated if the maximum number of generation reaches a predefined criteria, usually a sufficiently good fitness or in this case, a predefined maximum number of iterations (generations) is used. Different stopping conditions are tested such as 5, 10, 15, 25 and 30 and best solution could be obtained when number of generation is 30 for DE.

Crossover rate: Crossover rate (C_R) reflects the probability with which the trial vector inherits the actual vectors properties [32]. It is reported in the literature that if the C_R value is high, population diversity and convergence speed is improved[33]. Different levels of crossover rate (0.3, 0.5, 0.7, and 0.9) are tested. Best solution could be obtained when the C_R value is 0.9.

Mutation Factor: M is a mutation scaling factor of the difference vector (Equation 8). This parameter helps to control the evolving rate of the population. In the original DE algorithm it is reported that M value is chosen to be a value between 0 and 2. However, in the literature it is reported that small value of M leads to premature convergence and large value tends to slow down the search process. Hence in this research, mutation factor 0.5 is used for solving all 32 problems. A summary of the parameters chosen in this paper is presented in Appendix A.

4.3 Experimental Results

Thirty two test problems generated are solved for the proposed allocation procedure using differential evolution algorithm. The performances of the model are evaluated to find total

assembly line cost in a straight and U-shaped robotic assembly line. The proposed models are coded in C++ and the performances of DE are tested on Intel core i5 processor (2.3 GHz). The datasets evaluated are divided into two groups: small (problems with task size ranging between 25 and 70) and large size datasets (problems with task size ranging between 89 and 297) with different robot combinations. Table 7 shows the results obtained for the proposed DE algorithm using cost based and time based for straight robotic assembly line. Table 8 reports the results obtained using the two models for U-shaped robotic assembly line cost.

4.3.1 Experimental Results- straight robotic assembly line

Results of thirty two problems generated are compared for both the objectives in a straight robotic assembly line. The complete details of the results obtained by using the time based and cost based model for small size datasets (Problem No: 1 to 16) and for large size datasets (Problem No: 17 to 32) are presented in Table 7. Number of tasks and number of robots in the problem is presented in column 2. (For e.g. 25-3, read it as 25 tasks and 3 robot problem)The results reported are the best solution found using DE. From the table it is evident that cost based model is better in terms of minimizing the total assembly line cost when compared with time based model for both the groups of datasets and cycle time is better for time based data model when compared with the cost based data model. Assembly line cost evaluated using cost based model is lower when compared to assembly line cost obtained for time based model in a straight robotic assembly line. Percentage of cost saving obtained by using cost based model over the time based model is presented in the table along with percentage saving in cycle time using time based model is also presented. For straight line configuration, average cost saving by cost based model is 12.04% in case of small size problems and in case of large size problems the average cost saving is 11.57%. Average saving of cycle time by time based model is nearly 22% in case of small size problems and in case of large size problems cycle time improvement is more than 32%. From the above results, we can conclude that time based model is more appropriate as problem size increases. Figure 8 represents the saving potential in terms of percentage of assembly line cost for cost based model when compared with time based model in straight RAL. Two sets of the problem datasets are presented. For reader's clarity, authors have presented small datasets and large datasets in the same axis. In Figure 9, saving potential in terms of percentage of cycle time for time based model when compared with cost based model in straight RAL. Depending upon the priority of the management, the primary focus between time and cost could

vary at different time horizon. The appropriate model could be selected based on the priority of the management.

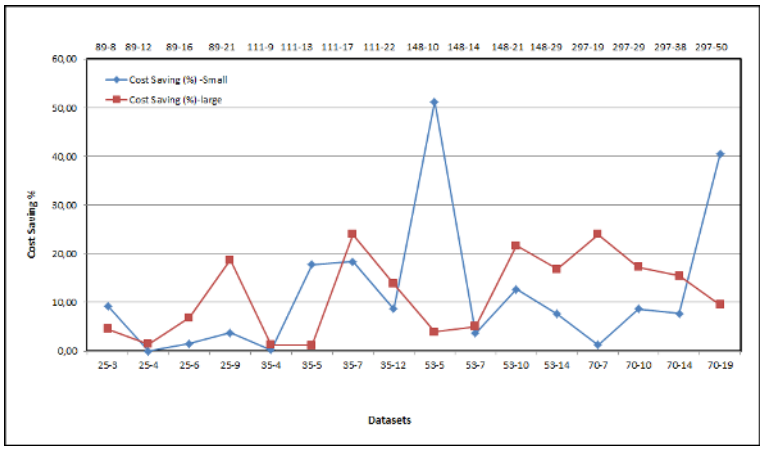


Figure 8 Cost saving percentage for straight RAL using cost based model

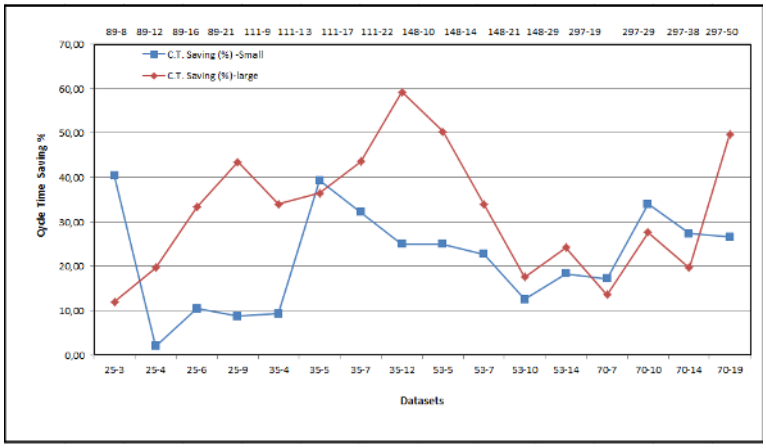


Figure 9 Cycle time saving percentage for straight RAL using time based model

4.3.2 Experimental Results- U-shaped robotic assembly line

Results of thirty two problems are compared for both the objectives (time and cost model) in a U-shaped robotic assembly line. The complete details of the results obtained by using the time based and cost based model for small size datasets (Problem No: 1 to 16) and for large size datasets (Problem No: 17 to 32) are presented in Table 8.Small size dataset problems contain problems with task sizes ranging from 25 to 70 tasks with different combination of robots and large size datasets contains problems with tasks sizes ranging from 89 to 297 tasks with different combination of robots. The results reported are the best solution found using DE. From the table it is evident that cost based model is better in terms of minimizing the total assembly line cost when compared with time based model for both the groups of datasets and cycle time is better

for time based data model when compared with the cost based data model for U-shaped robotic assembly line except for two datasets (53-7 and 148-14).

Assembly line cost evaluated using cost based model is lower when compared to assembly line cost obtained for time based model in a U-shaped robotic assembly line. Percentage of cost saving obtained by using cost based model over the time based model is presented in the table along with percentage saving in cycle time using time based model is also presented. In case of U-shaped RAL configuration average cost saving by cost based model when compared with time based model is 15.75% in case of small size problems and in case of large size problems the average cost saving is 10.72%. Average saving of cycle time by time based model when compared with cost based model is nearly 18.14% in case of small size problems and in case of large size problems cycle time improvement is more than 24.73%. From the above results, we can conclude that time based model is more appropriate as problem size increases. Figure 10 represents the saving potential in terms of percentage of assembly line cost for cost based model when compared with time based model in straight RAL. Two sets of the problem datasets are presented. In Figure 11, saving potential in terms of percentage of cycle time for time based model when compared with cost based model in straight RAL.

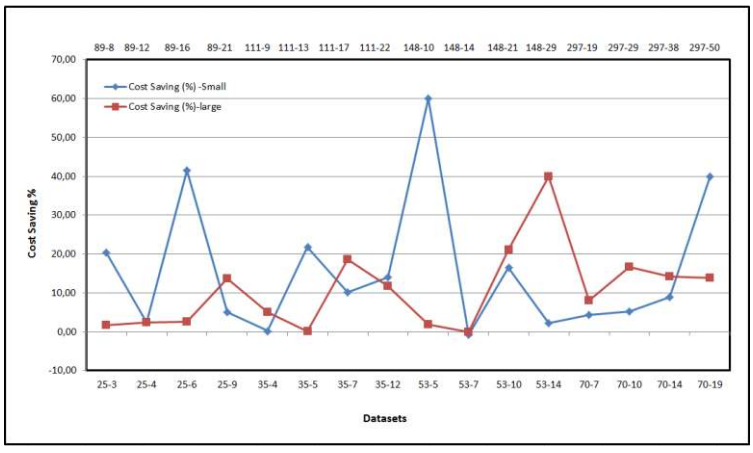


Figure 10 Cost saving percentage for U-shaped RAL by using cost based model

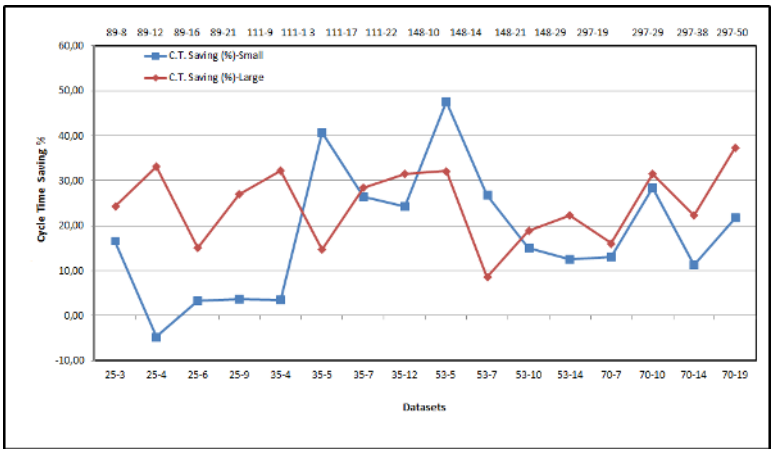


Figure 11 Cycle Time saving percentage for U-shaped RAL by using time based model

Table 7 Comparison of assembly line cost and cycle time for two models in straight RAL

Problem No:	Problem Dataset	Assembly Line Cost			Cycle Time			Problem No:	Problem Dataset	Assembly Line Cost			Cycle Time		
		By Cost Model	By Time Model	Cost Saving (%)	By Cost Model	By Time Model	C.T. Saving (%)			By Cost Model	By Time Model	Cost Saving (%)	By Cost Model	By Time Model	C.T. Saving (%)
1	25-3	**1218**	1331	9.28	706	**503**	40.36	17	89-8	**3124**	3264	4.48	516	**461**	11.93
2	25-4	**984**	984	0.00	299	**293**	2.05	18	89-12	**2863**	2904	1.43	383	**320**	19.69
3	25-6	**803**	815	1.49	221	**200**	10.50	19	89-16	**2472**	2641	6.84	292	**219**	33.33
4	25-9	**723**	750	3.73	124	**114**	8.77	20	89-21	**2288**	2716	18.71	244	**170**	43.53
5	35-4	**945**	947	0.21	374	**342**	9.36	21	111-9	**4231**	4284	1.25	698	**521**	33.97
6	35-5	**1317**	1551	17.77	464	**333**	39.34	22	111-13	**3335**	3375	1.20	438	**321**	36.45
7	35-7	**1273**	1507	18.38	279	**211**	32.23	23	111-17	**3299**	4088	23.92	349	**243**	43.62
8	35-12	**845**	918	8.64	130	**104**	25.00	24	111-22	**2794**	3179	13.78	293	**184**	59.24
9	53-5	**2230**	3371	51.17	561	**449**	24.94	25	148-10	**5613**	5832	3.90	881	**586**	50.34
10	53-7	**1768**	1832	3.62	362	**295**	22.71	26	148-14	**4220**	4431	5.00	561	**419**	33.89
11	53-10	**1666**	1877	12.67	252	**224**	12.50	27	148-21	**3722**	4528	21.66	321	**273**	17.58
12	53-14	**1299**	1398	7.62	168	**142**	18.31	28	148-29	**3744**	4374	16.83	236	**190**	24.21
13	70-7	**2319**	2348	1.25	504	**430**	17.21	29	297-19	**8311**	10301	23.94	675	**594**	13.64
14	70-10	**2173**	2360	8.61	351	**262**	33.97	30	297-29	**7570**	8876	17.25	503	**394**	27.66
15	70-14	**1966**	2118	7.73	247	**194**	27.32	31	297-38	**7598**	8771	15.44	365	**305**	19.67
16	70-19	**1718**	2413	40.45	176	**139**	26.62	32	297-50	**8320**	9112	9.52	331	**221**	49.77
	Total	**23247**	**26520**	**14.08**	**5218**	**4235**	**23.21**		**Total**	**73504**	**82676**	**12.48**	**7086**	**5421**	**30.71**
	Min.	**723**	**750**	**0.00**	**124**	**104**	**2.05**		**Min.**	**2288**	**2641**	**1.20**	**236**	**170**	**11.93**
	Max.	**2319**	**3371**	**51.17**	**706**	**503**	**40.36**		**Max.**	**8320**	**10301**	**23.94**	**881**	**594**	**59.24**
	Avg.	**1452.94**	**1657.50**	**12.04**	**326.13**	**264.69**	**21.95**		**Avg.**	**4594.00**	**5167.25**	**11.57**	**442.88**	**338.81**	**32.41**

*C.T. - Cycle Time

Table 8 Comparison of assembly line cost and cycle time for two models in U-shaped RAL

Problem No:	Problem Dataset	Assembly Line Cost			Cycle Time			Problem No:	Problem Dataset	Assembly Line Cost			Cycle Time		
		By Cost Model	By Time Model	Cost Saving (%)	By Cost Model	By Time Model	C.T. Saving (%)			By Cost Model	By Time Model	Cost Saving (%)	By Cost Model	By Time Model	C.T. Saving (%)
1	25-3	**1206**	1451	20.32	583	**500**	16.60	17	89-8	**3121**	3174	1.70	598	**481**	24.32
2	25-4	**965**	989	2.49	**303**	318	-4.72	18	89-12	**2850**	2921	2.49	425	**319**	33.23
3	25-6	**778**	1101	41.52	189	**183**	3.28	19	89-16	**2448**	2513	2.66	252	**219**	15.07
4	25-9	**704**	740	5.11	114	**110**	3.64	20	89-21	**2254**	2561	13.62	216	**170**	27.06
5	35-4	**945**	947	0.21	355	**343**	3.50	21	111-9	**4135**	4343	5.03	690	**522**	32.18
6	35-5	**1299**	1582	21.79	473	**336**	40.77	22	111-13	**3294**	3300	0.18	366	**319**	14.73
7	35-7	**1306**	1439	10.18	268	**212**	26.42	23	111-17	**3209**	3809	18.70	311	**242**	28.51
8	35-12	**795**	907	14.09	128	**103**	24.27	24	111-22	**2730**	3049	11.68	238	**181**	31.49
9	53-5	**2195**	3512	60.00	660	**447**	47.65	25	148-10	**5596**	5697	1.80	818	**619**	32.15
10	53-7	1739	**1725**	-0.81	359	**283**	26.86	26	148-14	4164	**4161**	-0.07	446	**411**	8.52
11	53-10	**1649**	1921	16.49	253	**220**	15.00	27	148-21	**3664**	4438	21.12	321	**270**	18.89
12	53-14	**1266**	1295	2.29	162	**144**	12.50	28	148-29	**3574**	5003	39.98	230	**188**	22.34
13	70-7	**2339**	2439	4.28	483	**427**	13.11	29	297-19	**8253**	8913	8.00	686	**591**	16.07
14	70-10	**2152**	2263	5.16	339	**264**	28.41	30	297-29	**7460**	8702	16.65	513	**390**	31.54
15	70-14	**1918**	2089	8.92	217	**195**	11.28	31	297-38	**7514**	8579	14.17	357	**292**	22.26
16	70-19	**1659**	2322	39.96	168	**138**	21.74	32	297-50	**8234**	9373	13.83	305	**222**	37.39
	Total	**22915**	**26722**	**16.61**	**5054**	**4233**	**290.31**		**Total**	**72500**	**80536**	**11.08**	**6772**	**5436**	**24.58**
	Min.	**704**	**740**	**-0.81**	**114**	**103**	**-4.72**		**Max.**	**2254**	**2513**	**-0.07**	**216**	**170**	**8.52**
	Max.	**2339**	**3512**	**60.00**	**660**	**500**	**47.65**		**Min.**	**8253**	**9373**	**39.98**	**818**	**619**	**37.39**
	Avg.	**1432.19**	**1670.13**	**15.75**	**315.88**	**263.94**	**18.14**		**Avg.**	**4531.25**	**5033.50**	**10.72**	**423.25**	**339.75**	**24.73**

*C.T.-cycle time

4.4 Evaluation (comparison) of straight and U-shaped RAL

The assembly line cost and cycle time obtained using cost based model and time based model for straight and U-shaped robotic assembly line are compared. Table 9 is formed by extracting the results from Table 7 and Table 8 obtained for minimizing total assembly line cost from straight and U-shaped robotic assembly line using cost based model results. The results indicate that total assembly line cost is very low for U-shaped robotic assembly line when compared to the total assembly line cost in straight robotic assembly line. Thirty out of thirty two datasets yielded lower assembly line cost for U-shaped robotic assembly line. Cost savings in terms of percentage by using U-shaped layout over straight line layout is presented in the table for both small and large size problems. U shaped layouts are better than straight line layout in both small size and large size problems and average cost savings by U shaped layout is around 1.6% when compared with straight line layout. Figure 12 presents the saving in cost by using U-shaped layout over straight layout for small size and large size problems. Assembly line cost is lower for U-shaped assembly line layout when compared with straight line layout due to the maximum resource utilization and more possibilities of task assignment in U-shaped layout.

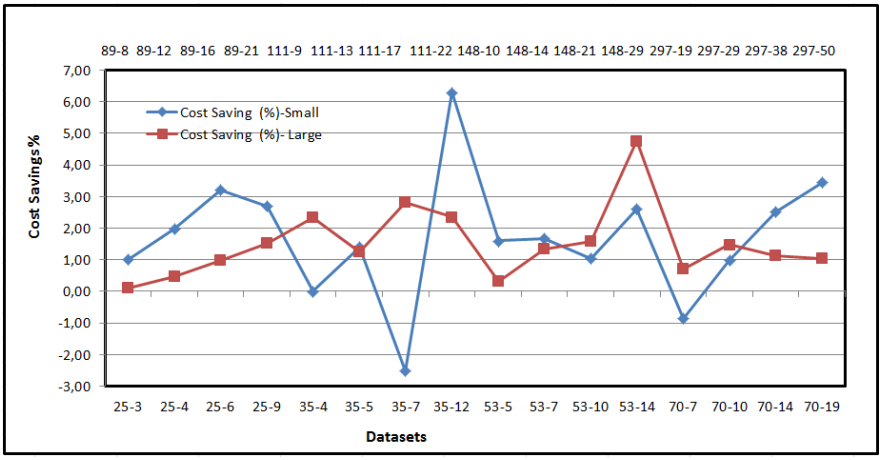


Figure 12 Cost saving percentage achieved using U-shaped RAL over straight RAL

Cycle time of both straight and U-shaped robotic assembly line obtained using time based model are extracted from Table 7 and Table 8 and the results are presented in Table 10. From Table 10, it is observed that cycle time of U-shaped robotic assembly line obtained using the time based model is lower than the cycle time for straight robotic assembly line problems for 21 out of 32 problems. The average percentage reduction in cycle time by U-shaped layout for the over straight layout is computed as 0.34%. It is concluded from this study that U-shaped robotic assembly line performs better than straight robotic assembly line for the objective of minimizing cycle time as well as minimizing total assembly line cost. Figure 13 presents the reduction in cycle time by using U-shaped layout over straight layout is presented for small size and large size problems.

Table 9 Comparison of assembly line cost - straight and U-shaped RAL

Problem No:	Problem Dataset	Assembly Line Cost			Problem No:	Problem Dataset	Assembly Line Cost		
		Straight RAL	U-Shaped RAL	Cost Saving %			Straight RAL	U-Shaped RAL	Cost Saving %
1	25-3	1218	**1206**	1.00	17	89-8	3124	**3121**	0.10
2	25-4	984	**965**	1.97	18	89-12	2863	**2850**	0.46
3	25-6	803	**778**	3.21	19	89-16	2472	**2448**	0.98
4	25-9	723	**704**	2.70	20	89-21	2288	**2254**	1.51
5	35-4	**945**	**945**	0.00	21	111-9	4231	**4135**	2.32
6	35-5	1317	**1299**	1.39	22	111-13	3335	**3294**	1.24
7	35-7	**1273**	1306	-2.53	23	111-17	3299	**3209**	2.80
8	35-12	845	**795**	6.29	24	111-22	2794	**2730**	2.34
9	53-5	2230	**2195**	1.59	25	148-10	5613	**5596**	0.30
10	53-7	1768	**1739**	1.67	26	148-14	4220	**4164**	1.34
11	53-10	1666	**1649**	1.03	27	148-21	3722	**3664**	1.58
12	53-14	1299	**1266**	2.61	28	148-29	3744	**3574**	4.76
13	70-7	**2319**	2339	-0.86	29	297-19	8311	**8253**	0.70
14	70-10	2173	**2152**	0.98	30	297-29	7570	**7460**	1.47
15	70-14	1966	**1918**	2.50	31	297-38	7598	**7514**	1.12
16	70-19	1718	**1659**	3.43	32	297-50	8320	**8234**	1.04
	Total	**23247**	**22915**	**-1.43**		**Total**	**73504**	**72500**	**-1,37**
	Min.	**723**	**704**	**-2.53**		**Min.**	**2288**	**2254**	**0.10**
	Max.	**2319**	**2339**	**6.29**		**Max.**	**8320**	**8253**	**4.76**
	Avg.	**1452.94**	**1432.19**	**1.69**		**Avg.**	**4594.44**	**4531.25**	**1.51**

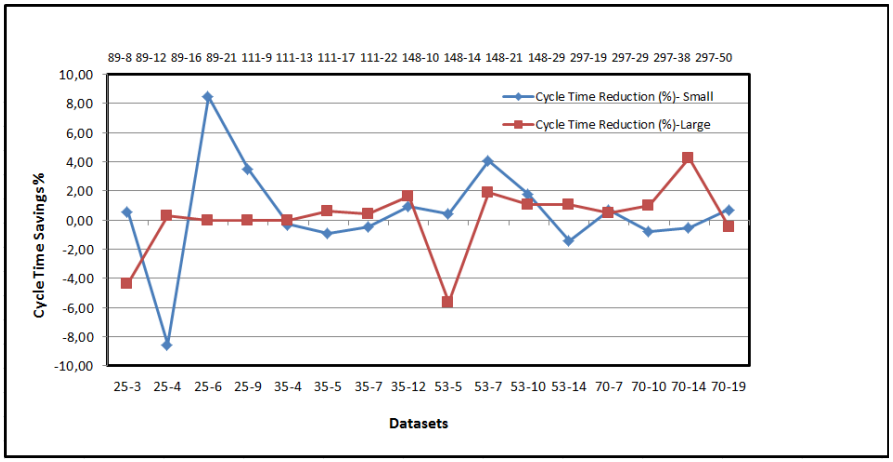


Figure 13 Cycle Time reduction percentage achieved using U-shaped RAL over straight RAL

Table 10 Comparison of cycle time - straight and U-shaped RAL

Problem No:	Problem Dataset	Cycle Time			Problem No:	Problem Dataset	Cycle Time		
		Straight RAL	U-Shaped RAL	Cycle Time Reduction %			Straight RAL	U-Shaped RAL	Cycle Time Reduction %
1	25-3	503	**500**	0.60	17	89-8	**461**	481	-4.34
2	25-4	**293**	318	-8.53	18	89-12	320	**319**	0.31
3	25-6	200	**183**	8.50	19	89-16	219	**219**	0.00
4	25-9	114	**110**	3.51	20	89-21	170	**170**	0.00
5	35-4	**342**	343	-0.29	21	111-9	521	**521**	0.00
6	35-5	**333**	336	-0.90	22	111-13	321	**319**	0.62
7	35-7	**211**	212	-0.47	23	111-17	243	**242**	0.41
8	35-12	104	**103**	0.96	24	111-22	184	**181**	1.63
9	53-5	449	**447**	0.45	25	148-10	**586**	619	-5.63
10	53-7	295	**283**	4.07	26	148-14	419	**411**	1.91
11	53-10	224	**220**	1.79	27	148-21	273	**270**	1.10
12	53-14	**142**	144	-1.41	28	148-29	190	**188**	1.05
13	70-7	430	**427**	0.70	29	297-19	594	**591**	0.51
14	70-10	**262**	264	-0.76	30	297-29	394	**390**	1.02
15	70-14	**194**	195	-0.52	31	297-38	305	**292**	4.26
16	70-19	139	**138**	0.72	32	297-50	**221**	222	-0.45
	Total	**4235**	**4223**	**-0.28**		**Total**	**5421**	**5435**	**0.26**
	Min.	**104**	**103**	**-8.53**		**Min.**	**170**	**170**	**-5.63**
	Max.	**503**	**503**	**8.50**		**Max.**	**594**	**619**	**4.26**
	Avg.	**264.69**	**264.69**	**0.52**		**Avg.**	**338.81**	**339.69**	**0.15**

For U-shaped robot assembly line, each task and any of its successor and/or predecessor can be allocated to the same workstation (tasks are allocated in U-shaped layout by repeatedly searching forward and backward through the precedence diagram) which gives more possible allocations and helps to reduce the workstation times and assembly line cost. Due to this, tasks from the both sides of the precedence diagram can be assigned to the same work station. In case of U-shaped layout, balancing the work load based on the demand by relocating the robots is easier. This provides more flexibility and adaptability for U-shaped layout and makes it an attractive layout when compared with straight line layout. Hence, by employing U-shaped robotic assembly line layout results in lower production cost and lower cycle time when compared with straight robotic assembly line layout.

For reader's clarity, Table 11 is presented to compare the average cycle time, average cost and average improvement in the cycle time and cost obtained using the proposed models for both straight line layout (SL) and U-shaped RAL problems. From the table it can be concluded that U-shaped layout performs better than the straight line layout. And in terms of average percentage comparison between models for both the layouts, it could be seen that time based model is capable of obtaining better solutions.

Table 11 Comparison of problem size, proposed models and RAL Configuration

Model		**Problem Size**			
		Small-SL	**Large-SL**	**Small-U**	**Large-U**
Time based	**Avg. Cost**	1657.50	5167.25	1670.13	5033.50
	Avg. C.T.	264.69	338.81	263.94	339.75
Avg. CT Improvement (%)		23.21	30.71	19.68	24.58
Cost based	**Avg. Cost**	1452.94	4594.00	1432.19	4531.25
	Avg. C.T.	326.13	442.88	315.88	423.25
Avg. Cost Improvement (%)		12.34	11.09	14.25	9.98
		Straight line RAL		**U-Shaped RAL**	

4.5 Computation time

Table 12 presents the average computation time for cost and time based model for both layouts considered in this paper. The quality of solution is given importance compared to the computation time. Average computation time is calculated and reported for each set of tasks.

When comparing the average computation time for cost based model, it can be seen that U-shaped layout needs more computation time than straight line layout for obtaining near optimal solutions. This is due to large search space in U-shaped layout and different possible combination of task allocation. Similarly for time based model, computation time is higher for U-shaped layout when compared with straight line layout. Further fine tuning of parameters can help to improve the robustness and computational efficiency of the proposed models.

Table 12 Comparison of average computation time

Problem Dataset	No: of Problems	Cost based model		Time based model	
		Straight RAL	U-shaped RAL	Straight RAL	U-shaped RAL
25	4	7	10	7	11
35	4	15	25	14	23
53	4	23	35	25	32
70	4	57	73	59	68
89	4	82	95	84	90
111	4	104	185	110	178
148	4	243	456	250	445
297	4	1235	1710	1240	1685

5. Managerial Insights and Conclusion

In this paper, the robotic assembly line balancing (RALB) problem with two different objectives viz., time and cost under two different configurations (straight line layout and U-shaped line layout) has been addressed. The work presented in this paper is an important addition to the literature where majority of the work so far focused only on robotic assembly line with the objective of minimizing cycle time. Two models (cost based model and time based model) are proposed to solve the robotic assembly line problem with an objective of minimizing cycle time and production cost. This problem falls under the category of NP-hard and hence solved using differential evolution (DE) algorithm. More than 30 data sets have been considered for evaluation under straight line and U-shaped configuration with objective of minimization of time and cost. Parametric study is conducted on selected problems to choose the efficient set of parameters for DE algorithm. From the experiments conducted, the following important managerial insights have been drawn.

- It is very important and critical to select suitable configuration (straight line or U-shaped) for assembly operations. This study can help managers or decision makers to choose suitable solution based on time and/or cost of performing the assembly operations using robots.
- From the experimental evaluation of performance of 32 problem sets, it can be observed that U-shaped assembly lines are more efficient (in 22 cases) both in terms of cost and time. However, in few cases, straight RALB is better than U shaped RALB. This clearly shows that decision makers/ managers need to evaluate the possible options clearly. This would help managers to choose appropriate configuration based on the floor space available etc.
- Managers can estimate the resources required under each configuration and corresponding performance. This study will also help in balancing the resources required and performance of the RALB
- It can also help in better planning and control of activities under different scenarios.

From the results presented in Table 7 and Table 8, it is noted that assembly line cost by cost based model is lowest compared to time based model for most of the problems considered in the evaluation under straight line and U-shaped configuration. Similar trend is observed in the case of time based model for cycle time. From the results given in Table 9 and 10, it is noted that assembly line cost and cycle time is better for U-shaped robotic assembly line when compared with straight robotic assembly line for most of the problems considered. These models can be strongly recommended to solve problem instances that occur in practice, regardless of the characteristics of the actual real-world problem.

In the future, different other efficient metaheuristics available in the literature can also be applied for solving the presently developed RALB problems and the performance of the proposed models from this paper can be used for the benchmark study. The models proposed in paper are for a single model, robotic assembly lines could be designed for assembly of mixed and multi models. Most of the literature published focused only on single objective optimization of RALB problems; there is a need to focus on multi-objective optimization of RALB problems.

References

1. Zhong RY, Dai Q, Qu T, Hu G, Huang GQ (2013) RFID-enabled real-time manufacturing execution system for mass-customization production. Robotics and Computer-Integrated Manufacturing 29 (2):283-292
2. Padrón M, de los A. Irizarry M, Resto P, Mejía HP (2009) A methodology for cost-oriented assembly line balancing problems. Journal of Manufacturing Technology Management 20 (8):1147-1165
3. Chica M, Bautista J, Cordón Ó, Damas S (2016) A multiobjective model and evolutionary algorithms for robust time and space assembly line balancing under uncertain demand. Omega 58:55-68
4. Amen M (2000) An exact method for cost-oriented assembly line balancing. International Journal of Production Economics 64 (1):187-195
5. Rosenberg O, Ziegler H (1992) A comparison of heuristic algorithms for cost-oriented assembly line balancing. Zeitschrift für Operations Research 36 (6):477-495
6. Hazir O, Delorme X, Dolgui A A Survey on Cost and Profit Oriented Assembly Line Balancing. In: 19th World Congress of The International Federation of Automatic Control, Cape Town, South Africa, 2014. vol 1. pp 6159-6167
7. Hahn R (1972) Produktionsplanung bei Linienfertigung. de Gruyter,
8. Steffen R (1977) Produktionsplanung bei Fließbandfertigung. Gabler, Wiesbaden,
9. Amen M (2000) Heuristic methods for cost-oriented assembly line balancing: A survey. International Journal of Production Economics 68 (1):1-14
10. Amen M (2001) Heuristic methods for cost-oriented assembly line balancing: A comparison on solution quality and computing time. International Journal of Production Economics 69 (3):255-264
11. Scholl A, Becker C (2005) A note on “An exact method for cost-oriented assembly line balancing”. International Journal of Production Economics 97 (3):343-352
12. Erel E, Sabuncuoglu I, Sekerci H (2005) Stochastic assembly line balancing using beam search. International Journal of Production Research 43 (7):1411-1426
13. Roshani A, Fattahi P, Roshani A, Salehi M, Roshani A (2012) Cost-oriented two-sided assembly line balancing problem: A simulated annealing approach. International Journal of Computer Integrated Manufacturing 25 (8):689-715
14. Hazır Ö, Delorme X, Dolgui A (2015) A review of cost and profit oriented line design and balancing problems and solution approaches. Annual Reviews in Control
15. Levitin G, Rubinovitz J, Shnits B (2006) A genetic algorithm for robotic assembly line balancing. European Journal of Operational Research 168 (3):811-825
16. Gao J, Sun L, Wang L, Gen M (2009) An efficient approach for type II robotic assembly line balancing problems. Computers & Industrial Engineering 56 (3):1065-1080
17. Nilakantan JM, Ponnambalam S, Jawahar N, Kanagaraj G (2015) Bio-inspired search algorithms to solve robotic assembly line balancing problems. Neural Computing and Applications 26 (6):1379-1393
18. Yoosefelahi A, Aminnayeri M, Mosadegh H, Ardakani HD (2012) Type II robotic assembly line balancing problem: An evolution strategies algorithm for a multi-objective model. Journal of Manufacturing Systems 31 (2):139-151
19. Nilakantan JM, Huang GQ, Ponnambalam S (2015) An investigation on minimizing cycle time and total energy consumption in robotic assembly line systems. Journal of Cleaner Production 90:311-325
20. Sivasankaran P, Shahabudeen P (2014) Literature review of assembly line balancing problems. The International Journal of Advanced Manufacturing Technology 73 (9-12):1665-1694
21. Mukund Nilakantan J, Ponnambalam S (2015) Robotic U-shaped assembly line balancing using particle swarm optimization. Engineering Optimization 48 (2):231-252
22. Scholl A, Becker C (2006) State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. European Journal of Operational Research 168 (3):666-693
23. Rashid MFF, Hutabarat W, Tiwari A (2012) A review on assembly sequence planning and assembly line balancing optimisation using soft computing approaches. The International Journal of Advanced Manufacturing Technology 59 (1-4):335-349

24. Storn R, Price K (1997) Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization 11 (4):341-359
25. Wang G-G, Hossein Gandomi A, Yang X-S, Hossein Alavi A (2014) A novel improved accelerated particle swarm optimization algorithm for global numerical optimization. Engineering Computations 31 (7):1198-1220
26. Nearchou AC (2008) Multi-objective balancing of assembly lines by population heuristics. International Journal of Production Research 46 (8):2275-2297
27. Karaboga N, Cetinkaya B (2004) Performance comparison of genetic and differential evolution algorithms for digital FIR filter design. In: Advances in information systems. Springer, pp 482-488
28. Ponnambalam S, Aravindan P, Naidu GM (2000) A multi-objective genetic algorithm for solving assembly line balancing problem. The International Journal of Advanced Manufacturing Technology 16 (5):341-352
29. Das S, Abraham A, Chakraborty UK, Konar A (2009) Differential evolution using a neighborhood-based mutation operator. Evolutionary Computation, IEEE Transactions on 13 (3):526-553
30. Davis L Applying adaptive algorithms to epistatic domains. In: IJCAI, 1985. pp 162-164
31. Scholl A (ed) (1995) Data of assembly line balancing problems. Darmstadt Technical University, Department of Business Administration, Economics and Law, Institute for Business Studies (BWL),
32. Feoktistov V (2006) Differential evolution. Springer,
33. Mohamed AW, Sabry HZ, Khorshid M (2012) An alternative differential evolution algorithm for global optimization. Journal of Advanced Research 3 (2):149-165

Appendix A- Parameters used in DE

Parameters
Population size: 25
Number of iterations: 30
Mutation factor: 0.5 Crossover rate: 0.9

Appendix B-Robot Cost data for small size datasets

Problem	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16	R17	R18	R19
11-4	1.1	1.2	1.25	1.3	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
25-3	1	1.5	1.2	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
25-4	1	1.25	1.15	1.2	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
25-6	1.05	0.95	1	1.2	1.5	1.3	--	--	--	--	--	--	--	--	--	--	--	--	--
25-9	1.3	1.5	1	1.2	0.95	1	1.1	1.25	1.1	--	--	--	--	--	--	--	--	--	--
35-4	1.05	0.95	1	1.2		--	--	--	--	--	--	--	--	--	--	--	--	--	--
35-5	1	1.5	0.8	1.2	0.87	--	--	--	--	--	--	--	--	--	--	--	--	--	--
35-7	1.35	0.95	1.1	1.25	1	1.5	1.15	--	--	--	--	--	--	--	--	--	--	--	--
35-12	1.15	1.25	0.825	0.95	1	1.5	1.35	1.1	1.2	0.875	1.15	0.975	--	--	--	--	--	--	
53-5	1	1.25	0.95	1.2	1.15	--	--	--	--	--	--	--	--	--	--	--	--	--	--
53-7	1.2	1.25	1.025	0.95	1.1	1.3	1.15						--	--	--	--	--	--	--
53-10	1.3	1.05	1.1	1.35	1.15	1.25	1.2	1.225	1.4	0.95			--	--	--	--	--	--	--
53-14	1.2	1.25	1.025	0.95	1.1	1.3	1.2	1.35	1.4	0.925	0.9	1.05	1.15	1	--	--	--	--	--
70-7	1	1.3	1.15	1.05	1.1	1.25	1.2					--	--	--	--	--	--	--	--
70-10	1.3	1.05	1.1	1.35	1.15	1.25	1.2	1.225	1.4	0.95	--	--	--	--	--	--	--	--	--
70-14	1.2	1.25	1.025	0.95	1.1	1.3	1.2	1.35	1.4	0.925	0.9	1.05	1.15	1					
70-19	1	0.82	0.9	1.05	1.3	1.4	1	1.1	0.95	1.225	0.95	1.2	1.35	1.25	1.325	1.15	1.25	1.3	0.8

*All the values are to be multiplied by 10^6 to get the actual data

Appendix C-Robot Cost data for large size datasets

Problem	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15	R16	R17	R18	R19	R20	R21	R22	R23	R24	R25
89-8	1	1.3	1.15	1.05	0.85	1.25	1.2	1.1	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
89-12	1.15	1.25	0.825	0.95	1	1.5	1.35	1.1	1.2	0.875	1.15	0.975	--	--	--	--	--	--	--	--	--	--	--	--	--
89-16	1.225	1.325	1.3	0.9	1.1	1.25	1.1	1.15	1	1.05	1.3	0.95	1.05	1.4	1.325	1.225	--	--	--	--	--	--	--	--	--
89-21	1.225	1.325	1.3	0.8	0.95	1.25	1.1	1.15	1	1.05	1.3	0.95	1.05	0.85	1.325	1.225	0.95	0.9	1.4	1.2	1.1	--	--	--	--
111-9	1.35	1.2	1.3	1.05	0.95	1.25	1.1	1.15	1.2	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
111-13	1.05	1.325	1.3	1.15	1.225	1.25	1.1	1.15	1	1.225	1.2	0.95	1.35	--	--	--	--	--	--	--	--	--	--	--	--
111-17	1.05	1.325	1.3	1.15	1.225	1.25	1.1	1.15	1	1.225	1.2	0.95	0.9	1.35	1	1.225	1.05	--	--	--	--	--	--	--	--
111-22	1	1.4	1.3	0.8	0.95	1.25	1	1.1	1	1	1.3	0.95	1	0.8	1.3	1.2	0.95	1.2	1.3	1.1	1.2	0.9	--	--	--
148-10	1.3	1.325	0.95	1.2	1.4	1.25	1.225	1.15	1	1.05	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
148-14	1.2	1.25	1	0.95	1	1.3	1.2	1.3	1.4	0.9	0.9	1	1.15	1	--	--	--	--	--	--	--	--	--	--	--
148-21	1.225	1.325	1.3	0.8	0.95	1.25	1.1	1.15	1	1.05	1.3	0.95	1.05	0.85	1.32	1.225	0.95	0.9	1.4	1.2	1.1	--	--	--	--
148-29:	1.225	1.325	1.3	0.8	0.95	1.25	1.25	1.1	1.15	1	1.05	1.3	0.95	1.05	0.85	1.325	1.225	0.95	1.4	1.2	1.1	1.225	1.325	1.3	0.8
297-19	1.225	1.325	1.3	0.8	0.95	1.25	1.1	1.15	1	1.05	1.3	0.95	1.05	0.85	1.325	1.225	0.95	0.9	1.4	--	--	--	--	--	--
297-29:	1.2	1.3	1.3	0.8	0.95	1.25	1	1.15	1	1	1.3	0.95	1	0.85	1.3	1.2	0.95	0.9	1.4	1.2	1.1	1.2	1.3	1.3	0.8
297-38:	1.2	1.3	1.3	1.05	1.3	1.25	1	1.1	1.1	1	1.3	1	1	0.9	1.3	1.2	0.9	1	1.4	1.2	1.1	1.2	1.3	1.3	0.8
297-50:	1.225	1.225	1.325	1.3	1.15	0.95	1.25	1.1	1.15	1	1.05	1.3	0.95	1.05	0.85	1.325	1.325	1.255	0.95	0.9	1.32	1.3	1.05	1.35	1.25

Problem	R26	R27	R28	R29	R30	R31	R32	R33	R34	R35	R36	R37	R38	R39	R40	R41	R42	R43	R44	R45	R46	R47	R48	R49	R50
89-8	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
89-12	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
89-16	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
89-21	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
111-9	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
111-13	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
111-17	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
111-22	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
148-10	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
148-14	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
148-21	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
148-29:	0.95	1.25	1.1	1.15	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
297-19	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
297-29:	0.95	1.25	1	1.1	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--
297-38:	0.95	1.25	1.1	1.15	1.2	1.3	1.3	1.15	0.95	1.25	1	1.15	1	--	--	--	--	--	--	--	--	--	--	--	--
297-50:	1.1	1.15	1.15	1.05	1.3	0.975	1.05	0.925	1.325	1.225	0.95	1.05	1.4	1.2	1.1	1.225	1.325	1.3	0.8	0.95	1.25	1.1	1.15	1.4	1.25

*All the values are to be multiplied by 10^6 to get the actual data