A Comparative Study of Assembly Line using GA with ACO

Dr. Umesh S. Mugale [*1], Dr. Vilas M. Nandedkar [2]

[*1]Professor in Mech. Engineering, VVP Institute of Engineering & Technology, Solapur, MS, India.

[2]Former Professor in Prod. Engineering., S.G.G.S.College of Engg. & Technology, Nanded, MS, India.

Abstract - Assembly line balancing problems arise whenever an assembly line is configured redesigned or adjusted. The decision problem of optimally balancing the assembly work or tasks among the stations with respect to some objective is known as the Assembly Line Balancing Problem (ALBP). ALBP tries to achieve the best compromise between labor, facility and resource requirements to satisfy a given volume of production. On the one hand, research has focused on developing effective and fast solution methods for exactly solving the simple assembly line balancing problem (SALBP). On the other hand, a number of real-world extensions of SALBP have been introduced but solved with straightforward and simple heuristics in many cases. Therefore, there is a lack of procedures for exactly solving such generalized ALBP.

Here, we have used Genetic Algorithm for simple assembly line balancing problem. The proposed Genetic Algorithm (GA) minimizes the cycle time for a given number of stations i.e. SALB-II. The proposed approach is compared with ACO and its performance analysis is tested on a set of test problems. The results show that the proposed approach performs well.

Keywords- Assembly line; Genetic Algorithm; ACO; cycle time; SALB-II.

I. INTRODUCTION

Assembly lines are flow-oriented production systems, which are typical in the industrial production of high quantity, standardized commodities as well as in low volume production of customized products.

An assembly line consists of work stations k = (1,….., m) arranged along a conveyor belt or a similar material handling equipment. The workpieces are consecutively launched down the line and are moved from station to station. At each station, certain operations are repeatedly performed regarding the cycle time. The decision problem of optimally partitioning the assembly work among the stations with respect to some objective is known as the Assembly Line Balancing Problem (ALBP).In a two-sided assembly line some tasks may be preferred to be performed at one side of the line, while others may be performed at either side (E) of the line.A pair of two directly facing stations is called as a mated-station and one of them calls the other a companion.

A precedence relation (i, j) means that task i must be finished before task j can be started and is represented as an arc in the precedence graph. An example of a precedence graph is given in Fig. 1.

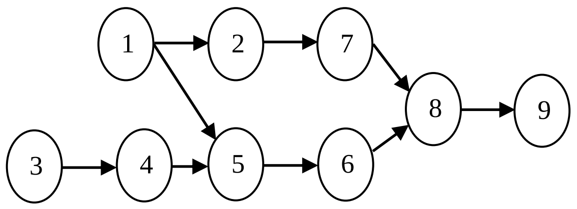


Fig. 1: Precedence graph

In this paper, we focus on SALBP-II; we have developed software in C++ language using Genetic Algorithm for assembly line balancing problem. The proposed Genetic Algorithm minimizes the cycle time for a given number of stations (SALBP-II). The proposed approach is illustrated with a problem, and its performance analysis is tested on a set of test problems.

The reminder of this paper is organized as, after this Introduction in Section 2, literature survey is given. In section 3, objectives of the problem are explained. Methodology used is explained in section 4. In Section 5, a problem to illustrate the proposed approach is solved & performance of the proposed approach is compared with ACO. Finally, some conclusions and future research directions are presented.

The SALB problem is categorized into two classes (Lee, Kim, & Kim, 2001); Type-I: minimization of the number of mated-stations (i.e., the line length) for a given cycle time, and Type-II: minimization of the cycle time for a given number of mated-stations.

II. LITERATURE SURVEY

Several studies on the mixed-model one-sided assembly line balancing problem have been reported in the literature (Dar-El & Cother, 1975; Erel &Gökcen, 1999; Gökcen& Erel, 1997, 1998; Jin & Wu, 2002; Karabati&Sayin, 2003; Macaskill, 1972; McMullen & Frazier, 1997, 1998; Merengo, Nava, &Pozetti, 1999; Simaria&Vilarinho, 2004; Thomopoulos, 1970; Vilarinho &Simaria, 2002, 2006). The detailed reviews of such studies have been given by Baybars (1986), Ghosh and Gagnon (1989), Erel and Sarin (1998), and more recently by Scholl and Becker (2006), Becker and Scholl (2006), Boysen and Scholl (2007). Although many researches have been done for one-sided ALB problems, considerably few researchers studied the TALB problem.

ALB has been an active field of research over more than half a century. This led to a massive body of literature covering plenty aspects of assembly line configuration. With regard to this tremendous academic effort in ALB, it is astounding that very few articles could be identified which explicitly deal with line balancing of real world assembly systems. In comparison to the 312 different research papers treated in the latest review articles of Scholl and Becker (2006), Becker and Scholl (2006) as well as Boysen et al. (2006a) this is less than 5%. This relation is another indicator for the aforementioned gap between the status of research and the requirements of real-world configuration problems. This work is intended to be a step towards closing this gap in the future.

In this work a problem with 12 tasks is considered from the literature Baykasoglu & Dereli (2009) in which ACO is used, in order to present the efficiency of the proposed GA.

III. OBJECTIVES

- Evaluation and Comparison of the Performance of the Developed Solution Procedures.
- To minimize the cycle time for the given number of stations.
- To improve the performance of the Assembly Line.
- To increase the line efficiency.
- To minimize the idle time on stations.
- To minimize the number of workers.
- To minimize the overall cost of assembly Line

IV. METHODOLOGY

4.1 Problem definition

An assembly line is designed to carry out a set of product models with similar production characteristics in any model sequence and model mix. Each model has its own set of task precedence relationships, and they can be combined into a single precedence diagram. The tasks in the combined precedence diagram for all models are performed on a set of mated-stations. A task i on a model m is performed in a certain time (t_{im}).

4.2 Genetic Algorithm (GA)

"A population containing a number of trial solutions each of which is evaluated (to yield fitness) and a new generation is created from the better of them. The process is continued through a number of generations with the aim that the population should evolve to contain an acceptable solution."

A genetic algorithm starts with an initial population of individuals (also called chromosomes or strings) representing different possible solutions to a problem. The population is maintained by the iterations of the algorithm, called generations. At each generation, the fitness of each individual is evaluated, and the individual is stochastically selected for the next generation based upon its fitness. New individuals called offspring are produced by two genetic operators, crossover and mutation. The offsprings are supposed to inherit the good attributes from their parents, so that the average quality of solutions becomes better than that in the previous population. This evolutionary process is repeated until some stopping criteria are met. The strength of a GA is the flexibility to adapt itself to changing optimization criteria and constraints. The performance of a GA is heavily influenced by the factors,such as the representation of the individuals; the decoding methods; the initial population; the selection scheme; and the choice of genetic operators and their combinations.

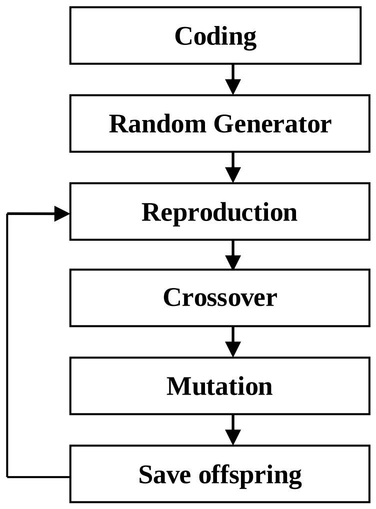


Fig. 2. Basic GA

4.3 Ant Colony Optimization (ACO)

ACO were first introduced by Dorigo et al.. Several sophisticated versions have been proposed to improve its performance. For a comprehensive review on ant algorithms, refer to Dorigo, Bonabeau and Theraulaz. The ant system has been applied to the job shop scheduling problem by Colorni et al. to the graph colouring problem by Costa and Hertz to the quadratic assignment problem by Maniezzo and to the vehicle routing problem by Bullnheimer et al.. The fundamental idea of ACO heuristics is based on the behavior of natural ants that succeed in finding the shortest paths from their nest to food sources by communicating via a collective memory that consists of pheromone trails. Due to ant's weak global perception of its environment, an ant moves essentially at random when no pheromone is available. However, it tends to follow a path with a high pheromone level when many ants move in a common area, which leads to an autocatalytic process. Finally, the ant does not choose its direction based on the level of pheromone exclusively, but also takes the proximity of the nest and of the food source respectively into account. This allows the discovery of new and potentially shorter paths.

V. RESULT AND DISCUSSIONS

A problem with 12 tasks is considered from the literature Baykasoglu & Dereli (2009) in which ACO is used, in order to present the efficiency of the proposed GA. Figure 3 shows the precedence diagram for 12 tasks whereas table 1 shows the tasks along with dependency & time.

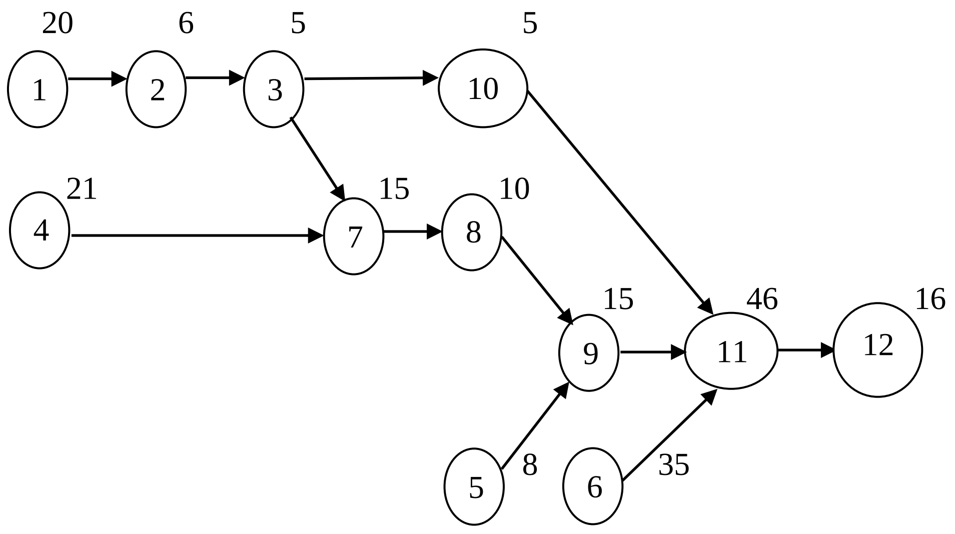


Figure 3: Precedence diagram for 12 tasks

Table 1: Tasks alongwith dependency and time

Task	Immediate Predecessor	Cycle Time in Sec.
1	-	20
2	1	6
3	2	5
4	-	21
5	-	8
6	-	35
7	3,4	15
8	7	10
9	5,8	15
10	3	5
11	6,9,10	46
12	11	16

Inputs: -

1. No. of Stations = 3

   Minimum Time = 202 / 3 = 67.33 Sec.

   Maximum Time = 202 / 2 = 101 Sec.

2. Cross over Probability = 0.8

   Mutation Probability = 0.1

3. Generate Random Number (1 to N)

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

4. Generate Initial Population :-

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

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

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

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

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

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

Table 2: Iteration number 1 for 12 tasks

Sr. No.	Y	F= 1/Y	Cum. A = F/F_A	Cum B	Rand. No.	Solution
1	272	0.00367	0.1613	0.1613	0.131	1
2	240	0.00416	0.1829	0.3442	0.352	3
3	273	0.00366	0.1609	0.5051	0.367	3
4	274	0.00364	0.1600	0.6651	0.695	5
5	271	0.00369	0.1622	0.8273	0.533	4
6	255	0.00392	0.1723	1	0.690	5
		F_A = 0.02274				

Reproduction	Crossover	Mutation
3-2-6-1-5-9-11-7-4-8-10-12	3-2-10-12-4-8-6-1-5-9-11-7	2-3-10-12-4-8-6-1-5-9-11-7
4-8-10-12-3-2-6-1-5-9-11-7	4-8-6-1-5-9-11-7-3-2-10-12	4-2-6-1-5-9-11-7-3-8-10-12
4-8-10-12-3-2-6-1-5-9-11-7	4-8-12-3-6-1-5-9-10-2-7-11	4-8-12-2-6-1-5-9-10-3-7-11
9-10-12-3-6-1-5-4-8-2-7-11	9-10-4-12-3-2-6-1-5-8-11-7	9-10-4-7-3-2-6-1-5-8-11-12
6-1-5-9-10-12-3-2-11-7-4-8	6-1-12-3-9-10-5-4-8-2-7-11	6-1-11-3-9-10-5-4-8-2-7-12
9-10-12-3-6-1-5-4-8-2-7-11	9-10-5-6-1-12-3-2-11-7-4-8	9-10-5-6-1-11-3-2-12-7-4-8

Table 3: Iteration number 2 for 12 tasks

Sr. No.	Y	F= 1/Y	Cum. A = F/F_A	Cum B	Rand. No.	Solution
1	238	0.00420	0.1656	0.1656	0.760	5
2	236	0.00423	0.1667	0.3323	0.332	2
3	228	0.00438	0.1727	0.5050	0.410	3
4	248	0.00403	0.1589	0.6639	0.552	4
5	241	0.00414	0.1632	0.8271	0.325	2
6	228	0.00438	0.1727	1	0.221	2
		F_A = 0.02536				

Reproduction	Crossover	Mutation
6-1-11-3-9-10-5-4-8-2-7-12	6-1-4-2-5-9-11-7-3-8-10-12	6-1-4-2-5-3-11-7-9-8-10-12
4-2-6-1-5-9-11-7-3-8-10-11	4-2-11-3-9-10-5-6-8-1-7-12	4-2-1-3-9-10-5-6-8-11-7-12
4-8-12-2-6-1-5-9-10-3-7-11	4-8-9-7-3-2-6-1-5-10-11-12	4-8-5-7-3-2-6-1-9-10-11-12
9-10-4-7-3-2-6-1-5-8-11-12	9-10-12-2-6-1-5-4-8-3-7-11	9-10-12-2-6-1-5-4-7-3-8-11
4-2-6-1-5-9-11-7-3-8-10-12	4-2-6-1-5-9-11-7-3-8-10-12	1-2-6-4-5-9-11-7-3-8-10-12
4-2-6-1-5-9-11-7-3-8-10-12	4-2-6-1-5-9-11-7-3-8-10-12	1-2-6-4-5-9-11-7-3-8-10-12

Table 4: Iteration number 3 for 12 tasks

Sr. No.	Y	F= 1/Y	Cum. A = F/F_A	Cum. B	Rand. No.	Solution
1	165	0.00606	0.2094	0.2094	0.473	3
2	196	0.00510	0.1762	0.3856	0.323	2
3	240	0.00416	0.1437	0.5293	0.448	3
4	272	0.00367	0.1268	0.6561	0.503	3
5	201	0.00497	0.1717	0.8278	0.336	2
6	201	0.00497	0.1717	1	0.880	6
		F_A = 0.02893				

Reproduction	Crossover	Mutation
4-8-5-7-3-2-6-1-9-10-11-12	4-2-1-3-9-10-5-6-8-11-7-12	4-2-1-3-7-10-5-6-8-11-9-12
4-2-1-3-9-10-5-6-8-11-7-12	4-2-5-7-3-8-6-1-9-10-11-12	4-2-5-1-3-8-6-7-9-10-11-12
4-8-5-7-3-2-6-1-9-10-11-12	4-8-5-7-3-2-6-1-9-10-11-12	4-1-5-7-3-2-6-8-9-10-11-12
4-8-5-7-3-2-6-1-9-10-11-12	4-8-5-7-3-2-6-1-9-10-11-12	4-1-5-7-3-2-6-8-9-10-11-12
4-2-1-3-9-10-5-6-8-11-7-12	4-2-6-1-5-9-11-7-3-8-10-12	1-2-6-4-5-9-11-7-3-8-10-12
1-2-6-4-5-9-11-7-3-8-10-12	1-2-4-3-9-10-5-6-8-11-7-12	1-2-4-3-7-10-5-6-8-11-9-12

Table 5: Iteration number 4 for 12 tasks

Sr. No.	Y	F= 1/Y	Cum. A = F/F_A	Cum B	Rand. No.	Solution
1	161	0.00621	0.1551	0.1551	0.076	1
2	140	0.00714	0.1783	0.3334	0.539	4
3	145	0.00689	0.1721	0.5055	0.172	2
4	145	0.00689	0.1721	0.6776	0.154	1
5	201	0.00497	0.1241	0.8017	0.075	1
6	126	0.00793	0.1981	1	0.929	6
		F_A = 0.04003				

Reproduction	Crossover	Mutation
4-2-1-3-7-10-5-6-8-11-9-12	4-2-5-7-3-1-6-8-9-10-11-12	1-2-5-7-3-4-6-8-9-10-11-12
4-1-5-7-3-2-6-8-9-10-11-12	4-1-2-3-7-10-5-6-8-11-9-12	4-1-2-3-7-10-5-6-8-9-11-12
4-2-5-1-3-8-6-7-9-10-11-12	4-2-1-3-7-10-5-6-8-11-9-12	4-1-2-3-7-10-5-6-8-11-9-12
4-2-1-3-7-10-5-6-8-11-9-12	4-2-5-1-3-8-6-7-9-10-11-12	4-1-5-2-3-8-6-7-9-10-11-12
4-2-1-3-7-10-5-6-8-11-9-12	4-2-1-3-7-10-5-6-8-11-9-12	4-1-2-3-7-10-5-6-8-11-9-12
1-2-4-3-7-10-5-6-8-11-9-12	1-2-4-3-7-10-5-6-8-11-9-12	1-2-4-3-7-10-5-6-8-9-11-12

Table 6: Iteration number 5 for 12 tasks

Sr. No.	Y	F= 1/Y	Cum. A = F/F_A	Cum B	Rand. No.	Solution
1	110	0.0090	0.1517	0.1517	0.644	4
2	80	0.0125	0.2107	0.3624	0.949	6
3	126	0.0079	0.1332	0.4956	0.495	3
4	105	0.0095	0.1602	0.6558	0.019	1
5	126	0.0079	0.1332	0.7890	0.387	1
6	80	0.0125	0.2107	1	0.560	4
		F_A = 0.0593				

Reproduction	Crossover	Mutation
4-1-5-2-3-8-6-7-9-10-11-12	4-1-2-3-7-10-5-6-8-9-11-12	1-4-2-3-7-10-5-6-8-9-11-12
1-2-4-3-7-12-5-6-8-9-11-12	1-2-5-4-3-8-6-7-9-10-11-12	1-2-5-4-3-7-6-8-9-10-11-12
4-1-2-3-7-10-5-6-8-11-9-12	4-1-5-7-3-2-6-8-9-10-11-12	4-1-5-2-3-7-6-8-9-10-11-12
1-2-5-7-3-4-6-8-9-10-11-12	1-2-4-3-7-10-5-6-8-11-9-12	1-2-4-3-7-10-5-6-8-9-11-12
1-2-5-7-3-4-6-8-9-10-11-12	1-2-5-4-3-8-6-7-9-10-11-12	1-2-5-4-3-7-6-8-9-10-11-12
4-1-5-2-3-8-6-7-9-10-11-12	4-1-5-7-3-2-6-8-9-10-11-12	4-1-5-2-3-7-6-8-9-10-11-12

Table 7: Iteration number 6 for 12 tasks

Sr. No.	Y	F= 1/Y	Cum. A = F/F_A	Cum B	Rand. No.	Solution
1	80	0.0125	0.1598	0.1598	0.554	4
2	75	0.0133	0.1700	0.3298	0.800	5
3	75	0.0133	0.1700	0.4998	0.958	6
4	80	0.0125	0.1598	0.6596	0.126	1
5	75	0.0133	0.1700	0.8296	0.803	5
6	75	0.0133	0.1700	1	0.761	5
		F_A = 0.0782				

Reproduction	Crossover	Mutation
1-2-4-3-7-10-5-6-8-9-11-12	1-2-5-4-3-7-6-8-9-10-11-12	1-2-3-4-5-7-6-8-9-10-11-12
1-2-5-4-3-7-6-8-9-10-11-12	1-2-4-3-7-10-5-6-8-9-11-12	1-2-4-3-7-10-6-5-8-9-11-12
4-1-5-2-3-7-6-8-9-10-11-12	4-1-2-3-7-10-5-6-8-9-11-12	4-1-2-3-7-10-6-5-8-9-11-12
1-4-2-3-7-10-5-6-8-9-11-12	1-4-5-2-3-7-6-8-9-10-11-12	1-2-5-4-3-7-6-8-9-10-11-12
1-2-5-4-3-7-6-8-9-10-11-12	1-2-5-4-3-7-6-8-9-10-11-12	1-2-3-4-5-7-6-8-9-10-11-12
1-2-5-4-3-7-6-8-9-10-11-12	1-2-5-4-3-7-6-8-9-10-11-12	1-2-3-4-5-7-6-8-9-10-11-12

Table 8: Iteration number 7 for 12 tasks

Sr. No.	Y	F= 1/Y	Cum. A = F/F_A	Cum B	Rand. No.	Solution
1	75	0.0133	0.1646	0.1646	0.423	3
2	72	0.0138	0.1707	0.3353	0.305	2
3	72	0.0138	0.1707	0.5060	0.425	3
4	75	0.0133	0.1646	0.6706	0.399	3
5	75	0.0133	0.1646	0.8352	0.774	5
6	75	0.0133	0.1646	1	0.358	3
		F_A = 0.0808				

Reproduction	Crossover	Mutation
4-1-2-3-7-10-6-5-8-9-11-12	4-1-2-3-7-10-6-5-8-9-11-12	1-2-4-3-7-10-6-5-8-9-11-12
1-2-4-3-7-10-6-5-8-9-11-12	1-2-4-3-7-10-6-5-8-9-11-12	1-2-3-4-7-10-6-5-8-9-11-12
4-1-2-3-7-10-6-5-8-9-11-12	4-1-2-3-7-10-6-5-8-9-11-12	1-4-2-3-7-10-6-5-8-9-11-12
4-1-2-3-7-10-6-5-8-9-11-12	4-1-2-3-7-10-6-5-8-9-11-12	1-4-2-3-7-10-6-5-8-9-11-12
1-2-3-4-5-7-6-8-9-10-11-12	1-2-4-3-7-10-6-5-8-9-11-12	1-2-3-4-7-10-6-5-8-9-11-12
4-1-2-3-7-10-6-5-8-9-11-12	4-1-3-2-5-7-6-8-9-10-11-12	4-1-2-3-5-7-6-8-9-10-11-12

Table 9: Iteration number 8 for 12 tasks

Sr. No.	Y	F= 1/Y	Cum. A = F/F_A	Cum B	Rand. No.	Solution
1	72	0.01388	0.1678	0.1678	0.208	2
2	72	0.01388	0.1678	0.3356	0.557	4
3	72	0.01388	0.1678	0.5034	0.621	4
4	72	0.01388	0.1678	0.6712	0.548	4
5	72	0.01388	0.1678	0.8390	0.146	1
6	75	0.01330	0.1608	1	0.655	4
		F_A = 0.0827				

Reproduction	Crossover	Mutation
1-2-3-4-7-10-6-5-8-9-11-12	1-2-3-4-7-10-6-5-8-9-11-12	1-2-3-4-7-10-6-5-8-9-11-12
1-4-2-3-7-10-6-5-8-9-11-12	1-4-2-3-7-10-6-5-8-9-11-12	1-4-2-3-7-10-6-5-8-9-11-12
1-4-2-3-7-10-6-5-8-9-11-12	1-4-2-3-7-10-6-5-8-9-11-12	1-4-2-3-7-10-6-5-8-9-11-12
1-4-2-3-7-10-6-5-8-9-11-12	1-4-2-3-7-10-6-5-8-9-11-12	1-4-2-3-7-10-6-5-8-9-11-12
1-4-2-3-7-10-6-5-8-9-11-12	1-4-2-3-7-10-6-5-8-9-11-12	1-4-2-3-7-10-6-5-8-9-11-12
1-4-2-3-7-10-6-5-8-9-11-12	1-4-2-3-7-10-6-5-8-9-11-12	1-4-2-3-7-10-6-5-8-9-11-12

Table 10: Iteration number 9 for 12 tasks

Sr. No.	**Y**	**F= 1/Y**	**Cum. A = F/F_A**	**Cum. B**	**Rand. No.**	**Solution**	**Reproduction**
1	72	0.0138	0.1666	0.1666	0.067	1	1-2-3-4-7-10-6-5-8-9-11-12
2	72	0.0138	0.1666	0.3332	0.830	5	1-4-2-3-7-10-6-5-8-9-11-12
3	72	0.0138	0.1666	0.4998	0.226	3	1-4-2-3-7-10-6-5-8-9-11-12
4	72	0.0138	0.1666	0.6664	0.924	6	1-4-2-3-7-10-6-5-8-9-11-12
5	72	0.0138	0.1666	0.8330	0.512	4	1-4-2-3-7-10-6-5-8-9-11-12
6	72	0.0138	0.1666	1	0.253	2	1-4-2-3-7-10-6-5-8-9-11-12
		F_A = 0.0828					

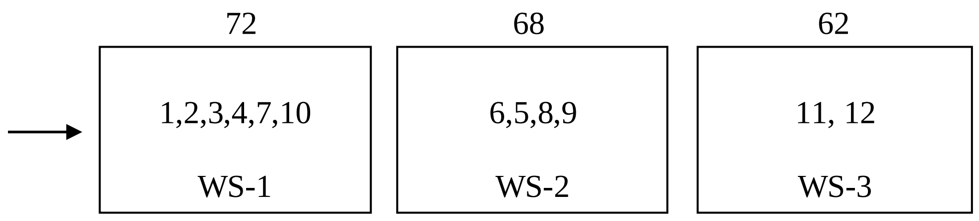


Figure 4: Task distributions using proposed GA

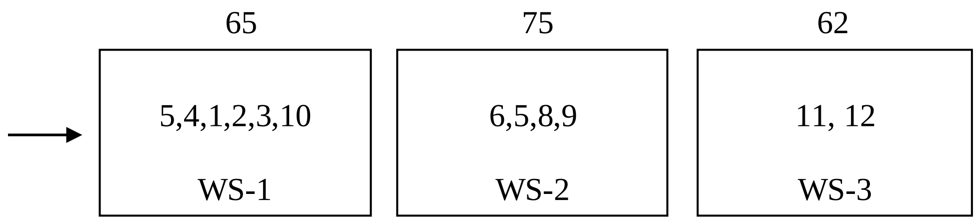
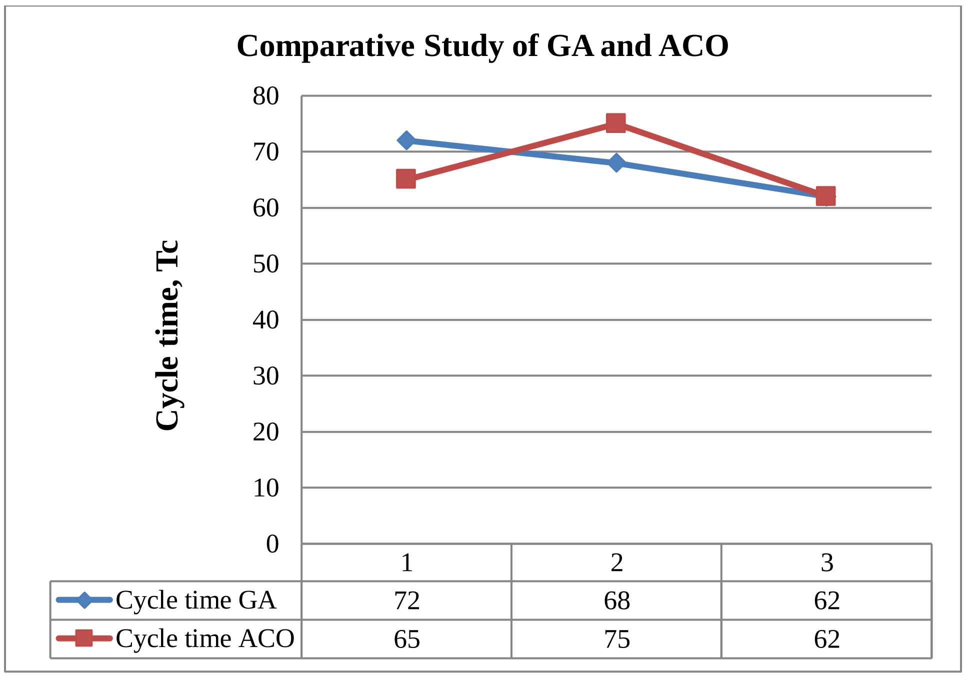


Figure 5: Task distributions using ACO (Baykasoglu & Dereli, 2009)



	1	2	3
Cycle time GA	72	68	62
Cycle time ACO	65	75	62

Figure 6: Graph showing comparative study of GA and ACO.

Figure 4 shows the distribution of tasks using the proposed GA in which the sequence of tasks to be performed is 1-2-3-4-7-10-6-5-8-9-11-12. By this sequence the the cycle time will be 72 Sec. which is less as compared to the example considered from the literature using ACO as shown in Figure 5. Again, it is observed that the ideal time at each workstation is minimum using proposed method and there is no violation of tasks. The comparative study of proposed GA and ACO is shown in Figure 6.

VI. CONCLUSION

Here, Genetic algorithm is used for solving SALBP-II. The algorithm is able quickly to search effective solutions for SALBP-II. The performance of the proposed algorithm is tested with several test problems from the literature. The proposed algorithm gives optimal solutions in short computational times. The cycle time obtained by proposed GA is 72 Sec. whereas by ACO is 75 Sec. It is concluded after this work that the proposed GA is good for solving ALBPs.

REFERENCES

[1] Baykasoglu, A., & Dereli, T., “Simple and U- Type Assembly Line Balancing by using an Ant Colony based Algorithm”, *Mathematical and Computational Applications*, 14 (1), 1-12, 2009.
[2] Dorigo, M. and Stutzle, T., “Ant Colony Optimization”, MIT Press, Cambridge, MA, 2004.
[3] Selvi, V., & Umarani, R., “Comparative Analysis of Ant Colony and Particle Swarm Optimization Techniques”, *International Journal of Computer Applications,* 5 (4), 0975 – 8887, 2010.
[4] Simaria, A. S., and Vilarinho, P. M., “2-ANTBAL: An ant colony optimization algorithm for balancing two-sided assembly lines”, *Computers & Industrial Engineering*, 56, 489–506, 2009.
[5] A. Scholl, C. Becker, “State-of-the-art exact and heuristic solution procedures for simple assembly line balancing”, *European Journal of Operational Research*, vol. 168, pp. 666–693, 2006.
[6] U. S. Mugae, V. M. Nandedkar, “Optimization of Assembly Line Balancing using Genetic Algorithm”, *International Journal of Advanced Manufacturing Systems,* 1(1), 1-5, 2010.
[7] C. Becker, A. Scholl, “A survey on problems and methods in generalized assembly line balancing”, *European Journal of Operational Research,* 168, 694–715, 2006.
[8] Y. Kim, Y.J. Kim & Y. Kim, “Genetic Algorithms for Assembly Line Balancing with Various Objectives”, *Computers Ind. Engg.* Vol. 30, No. 3, 397-409, 1996.