Novel Approach of Assembly Line Balancing by Hybrid Swarm Intelligence Approach

Mohit Kalia[1], Sourabh Dewedi 2,Varun Sharma[3]
[1,2,3] *Mechanical Engineering*
Indus International University, Una, Himachal, India.

Abstract **-** Assembly optimisation activities occur across development and production stages of manufacturing goods. Assembly Sequence Planning (ASP) and Assembly Line Balancing (ALB) problems are among the assembly optimisation. Both of these activities are classified as NP-hard. Several soft computing approaches using different techniques have been developed to solve ASP and ALB. Although these approaches do not guarantee the optimum solution, they have been successfully applied in many ASP and ALB optimisation works. This paper reported the survey on research in ASP and ALB that use soft computing approaches for the past 10 years. To be more specific, only Simple Assembly Line Balancing Problem (SALBP) is considered for ALB. The survey shows that three soft computing algorithms that frequently used to solve ASP and ALB are Genetic Algorithm, Ant Colony Optimisation and Particle Swarm Optimisation. Meanwhile, the research in ASP and ALB is also progressing to the next level by integration of assembly optimisation activities across product development stages. improve the productivity of the line and increases the quality of the product. To find an optimal solution for Robotic Assembly Line Balancing (rALB) problem we will have to assign robots to stations in a balanced manner to perform activities. The main objective of this work is to minimize the cycle time and maximize the production rate of the line. A particle swarm optimization method is proposed to find optimum solution for the rALB problem. Results obtained using the PSO are further improved by using a local exchange procedure. Performance of the proposed method is tested on benchmark rALB problems. The results of PSO are found to be better than the methods reported in the literature and it produce consistent results.

Keywords **-** Line balancing, Assembly Line Balancing, Tasks and Precedence, Earth Mover's Distance

I. INTRODUCTION

The concept of Assembly Line Balancing Line balancing is about arranging a production line so that there is an even flow of production from one work station to the next. Line balancing also a successful tool to reduce bottleneck by balancing the task time of each work station so that there is no delays and nobody is overburden with their task [1] [2]. An assembly line is a manufacturing process in which parts are added to a product in a sequential manner using optimally planned logistics to create a finished product in the fastest possible way. It is a flow-oriented production system where the productive units performing the operations, referred to as stations, are aligned in a serial manner. Assembly Line Balancing, or simply Line Balancing (LB), is the problem of assigning operations to workstations along an assembly line, in such a way that the assignment be optimal in some sense [3] [4]. Furthermore, an assembly line can also be defined as a system which is formed by arranging workstations along a line. At these workstations, work pieces can be transferred by using labor force as well as equipment, and tasks are assembled taking into consideration precedence constraints and cycle time. The decision problem of optimally balancing the assembly work among the workstations is pointed out by M.Baskak (2008) as the assembly line balancing problem [4]. The objectives of balancing and optimization of assembly lines is twofold, either cost minimization or profit maximization [11]. The various characteristics for purpose of balancing and optimization are as follows:

- Number and variety of product.
- Line control
- Variability of task times
- Line layout
- Parallelization of assembly work
- Equipment and processing alternatives
- Assignment restrictions
- Worker productivity

A. Terms and Concepts - Assembly lines are an integral part of any manufacturing process to complete a product. These lines are fundamentally flow lines in which the work moves from one station to another with addition of material at each station. Under ideal conditions, the elemental time at each station has to be same. This statement is quiet contrary in the real world situation since, most of assembly lines are manually operated giving rise to different timings at each station as per the capacity of the man allocated to the station [6].

i). Assembly: As described by (Scholl), is a manufacturing process that develops a work-in-progress workpiece into finished product by sequential attachment of parts. Parts are the atomic physical inputs to the assembly process, each of which is typically standardized and interchangeable with other parts of the same type [8]. A subassembly is a collection of parts that are attached to one another, prior to fastening to the workpiece.

ii). Tasks and Precedence: The work performed during assembly is portioned into the smallest possible indivisible

operations, or tasks, each of which requires an associated task time to complete. The sequence in which tasks are performed may be constrained such that some tasks must be done before another task begins, due to the physical architecture of the workpiece, safety reasons, or other causes. Precedence relationships between two individual tasks are used to codify these constraints, with the task that must come first labelled the predecessor and the later task called the successor. The set of all binary precedence relationships between task pairs may be represented as a precedence graph, by first drawing each task as a node and then drawing directed arcs pointing away from each predecessor task towards its successor. An example precedence graph is shown in Figure 1. The precedence graph must be a cyclic, as no task may be considered a predecessor to itself. It is not required for all nodes in the graph to communicate, as disconnected subgraphs indicate that the corresponding tasks are precedence independent from one another. Nor it is required to draw indirect precedence relationships on the graph. For example, in Figure 1, task 2 is a predecessor to task 7, but this relationship is implicit by considering the predecessor relationships of task 4.

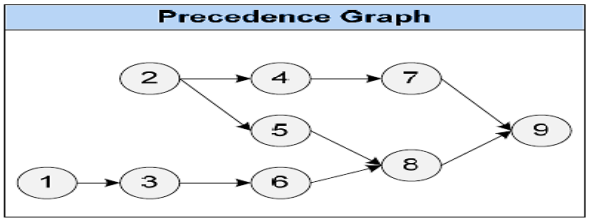

Figure 1: Precedence Graph [3]

Alternatively, precedence relationships may be collected in the form of a precedence matrix. Each task may be arbitrarily assigned an indexing number, 1 to n, where n is the total number of tasks. The rows of the n-x-n matrix index to predecessor tasks and successors are indexed to columns, allowing one matrix element for each possible precedence relationship. The matrix is constructed by placing a 1 in each matrix element for which a precedence relationship exists, and a 0 if not. An example of a precedence matrix is shown in Figure 2, containing the same precedence information as in Figure 1.

	1	2	3	4	5	6	7	8	9
1	-	0	1	0	0	0	0	0	0
2	0	-	0	1	1	0	0	0	0
3	0	0	-	0	0	1	0	0	0
4	0	0	0	-	0	0	1	0	0
5	0	0	0	0	-	0	0	1	0
6	0	0	0	0	0	-	0	1	0
7	0	0	0	0	0	0	-	0	1
8	0	0	0	0	0	0	0	-	1
9	0	0	0	0	0	0	0	0	-

Figure 2: Precedence Matrix [3]

Note that there are many indirect precedence relationships that are not tracked in the above example precedence graph and precedence matrix. Instead only immediate precedence relationships are shown, i.e. the minimal set of arcs necessary to constrain the acyclic graph. For example, task 1 is a predecessor for tasks 3, 6, 8, and 9, but only the relationship to task 3 is immediate. All indirect precedence relationships may be derived from the set of direct precedence relations, if desired.

iii). Assembly Lines, Stations, and Workers: An assembly line is a type of assembly process, in which a conveyor or similar material handling equipment moves evenly spaced workpieces from the beginning of the assembly process to the end. The conveyance path is segmented according to this spacing into a series of consecutive stations, such that there is one workpiece in each station [5]. Each station is given a subset of tasks to complete, and the requisite parts, tooling, and other needs in order to complete those tasks, in addition to a worker to provide necessary manpower. Fixed pace assembly lines convey workpieces at a steady rate from one station to the next, resulting in a constant cycle time for each station to complete work on the current workpiece before the conveyor moves it to the next station.

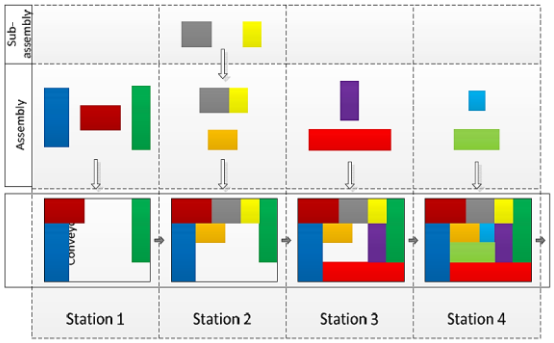

Figure 3: An example of an assembly line [3]

An example of an assembly line is shown in Figure 3. In this pictogram, each block represents a part. At each station a worker picks the parts, optionally sub-assembles some of them, and fastens them into the workpiece upon the conveyor [3] [7]. Assembly lines were originally constructed for mass production of standardized assembly products, to increase average worker productivity and overall throughput by leveraging labour specialization along the line (Dar-El and Shtub). Modern assembly lines designed for make-to-order and mass customization production permit fast and flexible responses to customer demand, but are associated with significant5automation and facility capital costs. Successful assembly line planning is critical to engineering a cost-effective production process.

B. The Assembly Line Balancing Problem - The assembly line balancing problem (ALB) is a production planning problem concerned with allocating tasks to the stations on the assembly line, first proposed [13] and formulated as a mathematical programming problem in 1955 by (Salveson). A solution to the ALB is a set of decisions that determine which tasks are assigned to each

station. (Scholl) provides a thorough modern review of assembly lines and the ALB.

C. Need of using ALB - In real world, assembly line balancing relate to finite set of work elements, and each element having relationship of processing time and precedence. Line balancing is an attempt to equal amount work to each work station to achieve the desired efficiency by concentrating the factors like minimizing work stations, minimizing work load variations and cycle time minimization. Therefore, assembly line balancing attracted the attention of researchers who tried to support practical configuration planning by suited optimization models. In spite of the great amount of extensions of basic assembly line balancing there remains a gap between requirements of real configuration problems and the status of research [14]. Assembly line balancing in automobile industry is of big interest because of complex nature of problem, deviating situation including objectives. The multi model assembly line (MALB) has many variants for purpose of consideration. Also, the manual assemble line lead to either blocking or starvations because of non uniforms cycle time at each station. In this case study, the problem of reconfiguration is not redesigning the line which may include retention of the workers, sweeping changes in layout design or reallocating space for storage. By reconfiguration over here, it is assumed that line efficiency is to be enhanced either by reducing the no of stations or by regrouping the cycle times which will maximize the utilization and reduce blocking-starvation problem as far as possible.

D. Generalizations of ALB - For any valid ALB solution, the following minimal set of constraints must be satisfied:

1. All tasks must be assigned to some station, such that the workpiece is finished upon exiting the final station.
2. All precedence relationships must be satisfied. Classically this constraint is enforced by ensuring that no task is assigned to an earlier station than one of its predecessor tasks.
3. The sum of task times at each station cannot exceed the cycle time.

Using the terminology of (Baybars, A Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem) survey, Salveson's initial formulation is known as the Simple Assembly Line Balancing Problem (sALB), as it features a number of simplifying assumptions [12] [13]:

- Mass-production of one homogenous product.
- All tasks are processed in a predetermined mode (no processing alternatives exist).
- Paced line with a fixed common cycle time according to a desired output quantity.
- The line is considered to be serial with no feeder lines or parallel elements.
- The processing sequence of tasks is subject to precedence restrictions.
- Deterministic (and integral) task times.
- No assignment restrictions of tasks besides precedence constraints.
- A task cannot be split among two or more stations.
- All stations are equally equipped with respect to machines and workers.

Many industrial environments do not conform to these assumptions, motivating a vast body of research addressing specific manufacturing conditions that require relaxation of one or more assumptions. Though extensive research has been--and continues to be--published relating to ALB, the field is marked by increasingly divergent extensions to the core problem. Some authors have sought to nest ALB within a larger framework of engineering decision problems such as facility design, equipment selection, production scheduling, and logistics. Others have developed focused ALB techniques that conform to specific characteristics of real-world ALB problems. Taken together, these generalizations cover a very wide, but sparse domain, as there are a huge number of16problem characteristic combinations possible, and relatively few problem extension approaches amenable to simultaneous application.

i). Single-Model: In early times assembly lines were used in high level production of a single product. But now the products will attract customers without any difference and allows the profitable utilization of Assembly Lines. An advanced technology of production which enables the automated setup of operations and it is negotiated time and money [5] [10]. Once the product is assembled in the same line and it won't variant the setup or significant setup and it's time that is used, this assembly system is called as Single Model Line.

ii). Mixed-Model: In this model the setup time between the models would be decreased sufficiently and enough to be ignored. So this internal mixed model determines the assembled on the same line. And the type of assembly line in which workers work in different models of a product in the same assembly line is called Mixed Assembly Line.

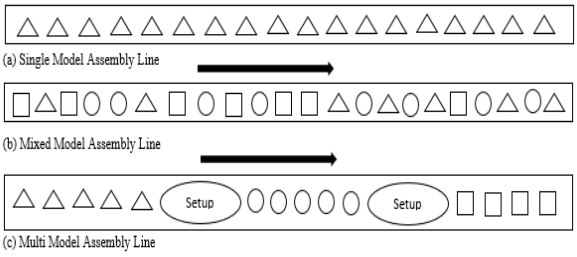

Figure 4: Assembly lines for single and multiple products

iii). Multi-Model: In this model the uniformity of the assembled products and the production system is not that much sufficient to accept the enabling of the product and the production levels [7]. To reduce the time and money this assembly is arranged in batches, and this allows the short term lot-sizing issues which made in groups of the

models to batches and the result will be on the assembly levels.

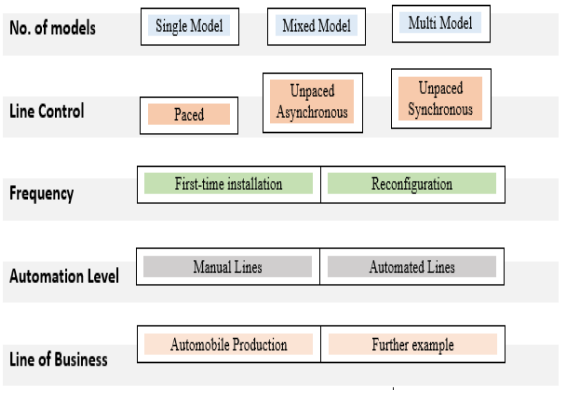

Figure 5: Investigated kinds of ALB

E. Generalized Assembly Line Balancing Problems (GALBP) - In generalized assembly line balancing problems one or more assumptions of the simple case are relaxed (Bay bars, 1986; Scholl and Becker, 2006).Some common GALBP are:

i). U-Shaped Assembly Line Balancing Problem (UALBP): This is U-shaped lines. This configuration is considered to be more flexible. It allows more possibilities on how to assign tasks to workstations. The reason for this is that tasks can be assigned when either its predecessors have already been assigned, whereas with serial lines a task can be assigned only when its predecessors have been assigned. Its variants are: UALBP-1, UALBP-2 and UALBP-E respectively.

ii). Mixed-model Assembly Line Balancing Problem (MALBP): Karabati and Sayin (2003), Ponnambalam et al. (2003), Spina et al. (2003), Bukchin and Rabinowitch (2006) have addressed MALBP in their works [7] [14] . Different models of the same product are inter-mixed. On the same line these products are to be assembled. So, the sequence of different models has to be determined.MALPB-1, MALBP-2 and MALBP-E are the different type's present here.

iii). Robotic Assembly Line Balancing Problem (RALBP): Robotic line is considered here. Problem considers the assignment of set of tasks and the set of robots to workstations (Rubinovitz and Bukchin, 1993).

iv). Multi-objective Assembly Line Balancing Problem (MOALBP): Several optimization objectives are considered simultaneously. Agpak and Gokcen (2005) deal with a problem that seeks to minimize both the number of workstations and the total assembling cost or the amount of resources. Most GALBP are of multi-objective nature.

II. RELATED WORK

Jordi Pereira, et.al [1] studied an assembly line balancing problem with uncertainty on the task times. In order to deal with the uncertainty, a robust formulation to handle changes in the operation times is put forward. In order to solve the problem, several lower bounds, dominance rules and an enumeration procedure are proposed. These methods are tested in a computational experiment using different instances derived from the literature and then compared to similar previous approaches. A. Dolgui, et.al [2] explained the problems, approaches and analytical models on assembly line balancing that deal explicitly with cost and profit oriented objectives are analysed. This survey paper serves to identify and work on open problems that have wide practical applications. The conclusions derived might give insights in developing decision support systems (DSS) in planning profitable or cost efficient assembly lines. Naveen Kumar, et.al [3] presented the reviews of different works in the area of assembly line balancing and tries to find out latest developments and trends available in industries in order to minimize the total equipment cost and number of workstations. M. Baskak [4] explained two heuristic assembly line balancing techniques known as the "Ranked Positional Weight Technique", developed by Helgeson and Birnie, and the "Probabilistic Line Balancing Technique", developed by El-Sayed and Boucher, were applied to solve the problem of multi-model assembly line balancing in a clothing company for two models. Information about definitions and solution methods related to assembly line balancing problems was given. The aim of this article was the comparison of the efficiencies of two different procedures applied for the first time to solve line balancing in a clothing company. By using both methods, different restrictions are taken into consideration and two different line balancing results are reached. The balancing results are compared with each other. K. Rengarajan, et.al [5] dealt with mixed-model assembly line balancing for n models, and uses a classical genetic algorithm approach to minimize the number of workstations. We also incorporated a hybrid genetic algorithm approach that used the solution from the modified ranked positional method for the initial solution to reduce the search space within the global space, thereby reducing search time. Several examples illustrate the approach. The software used for programming is C++ language. S.D Lapierre, et.al [6] presented a case study where a practical balancing problem for an assembly line of appliances with two sides and two different heights was solved with an enhanced priority-based heuristic. The researchers have shown how to adapt such heuristic to account for the practical aspects of industrial applications. The experts have also shown good use of logic and randomness in the algorithm is the key to allow the heuristic to find good solutions. In order to speed up its implementation and to facilitate software maintenance, we have implemented the heuristic on MsAccess97. Selcuk Karabat, et.al [7] considered the assembly line balancing problem in a mixed-model line which is operated under a cyclic sequencing approach. The experts specifically studied the problem in an assembly line environment with synchronous transfer of parts between the stations. The researchers formulated the assembly line balancing problem with the objective of minimizing total cycle time by incorporating the cyclic sequencing information. The study has shown that the

solution of a mathematical model that combines multiple models into a single one by adding up operation times constitutes a lower bound for this formulation. As an approximate solution to the original problem, the researchers also proposed an alternative formulation that suggests to minimize the maximum sub cycle time. A simple heuristic approach for this alternative problem was also developed. C. Becker, et.al [8] proposed that the Assembly Line Balancing Problem consist the finding of a feasible line balance, i.e., an assignment of each task to a station such that the precedence constraints and further restrictions are fulfilled. A usual surrogate objective consists in maximizing the line utilization which is measured by the line efficiency as the productive fraction of the line's total operating time and directly depends on the cycle time c and the number of stations. The survey reveals that assembly line balancing research which traditionally was focused upon simple problems (SALBP) has recently evolved towards
formulating and solving generalized problems (GALBP) with different additional characteristics such as cost functions, equipment selection, paralleling, U-shaped line layout and mixed-model production. T.O, Lee, et.al [9] considered two-sided (left- and right-side) assembly lines that were often used in assembling large-sized products, such as trucks and buses. A large number of exact algorithms and heuristics have been proposed to balance one-sided assembly lines. However, little attention has been paid to balancing the two-sided lines. An efficient assignment procedure was developed for two-sided assembly line balancing problems. A special emphasis is placed on maximizing work relatedness and maximizing work slackness, which were of practical significance especially in two-sided lines. The experts first investigated the characteristics of two-sided lines and define new measures for the balancing. Then, a group assignment procedure, which assigns a group of tasks at a time rather than a unit task, is designed. Experiments are carried out to demonstrate the performance of the proposed method. The results show that our procedure is promising in the solution quality. B. J., Carnahan, et.al [10] presented a study that involved three line balancing heuristics that incorporate physical demand criteria to solve the problem of finding assembly line balances that consider both the time and physical demands of the assembly tasks: a ranking heuristic, a combinatorial genetic algorithm, and a problem space genetic algorithm. Each heuristic was tested using 100 assembly line balancing problems. Incorporating physical demands using these algorithms does impact the assembly line configuration. Results indicated that the problem space genetic algorithm was the most adept at finding line balances that minimized cycle time and physical workload placed upon participants. Benefits of using this approach in manufacturing environments are discussed. A, Pozzetti, et.al [11] analysed some typical problems of manual, mixed-model assembly lines. In particular, it presents new balancing and production sequencing methodologies which pursue the following common goals: (1) minimizing the rate of incomplete jobs (in paced lines and in moving lines) or the probability of blocking/starvation events; (2) reducing WIP. The balancing methodology also aims at minimizing the number of stations on the line; the sequencing technique also provides a uniform parts usage, which is a typical goal in just in time production systems. I, Baybars [12] discussed the development of the simple assembly line balancing
problem (SALBP); modifications and generalizations over time; present alternate 0-1 programming formulations and a general integer programming formulation of the problem; discuss other well-known problems related to SALBP; describe and comment on a number of exact ((i.e. optimum-seeking) methods; and present a summary of the reported experiences. All models discussed here are deterministic (i.e., all input parameters are assumed to be known with certainty) and all the algorithms discussed are exact. The problem is termed "simple" in the sense that no "mixed-models," "subassembly lines," "zoning restrictions," etc. are considered. M.E Salveson [13] presented the application of fuzzy logic in balancing a single model tricycle assembly line. MATLAB simulation software was used in the analysis of the primary and secondary data obtained from the assembly line under study. Results obtained from the study show that the efficiency of the line increased from 88.1% to 92.4%. The total idle time was also reduced by 56.5%. This indicates an improvement in the efficiency of the line, reduction of bottleneck, and even distribution of tasks along the line for the company under study.

III. THE PROPOSED METHOD

A. Proposed Methodology - In proposed work Genetic Algorithm (G.A) and Particle Swarm Optimization algorithm is used for the better optimization results. Genetic algorithm is a meta-heuristic algorithm which is based on the gene and their operation. In the genetic algorithm all the process is based on the selection, cross-over and mutation operation for the optimal results. The optimization is based on the fitness value of the genes. This algorithm supports the local optimization process which is not enough to get the effective results. To overcome this issue the hybrid approach is proposed in the present work. The Particle swarm Optimization algorithm is a meta-heuristic algorithm which is based on the behavior of swarms. This algorithm is used to solve the complex problem to get the optimal results. PSO supports the Global optimization feature and gives the solution of the problem which is globally best. In the present work, PSO and G.A work parallel for better and optimal solution because both have different feature of optimization. In the below given section we explain the Genetic Algorithm (G.A) and Particle Swarm optimization (PSO) with algorithm and their flow chart. The flow chart of explains the step by step working and algorithm represents the technical implementation of the algorithms.

Step 1: Initialize the Load/Power.
Step 2: Initialize the generator Load_ Power.
Step 3: Allocate the generators and calculate the cost.
Step 4: Apply the PSO for optimization.
Step 5: If output of PSO is optimized then check the convergence otherwise Genetic algorithm starts it working with the following steps.
(a) Initialize the chromosomes.
(b) Cross over between chromosomes.
(c) Apply Roulette Selection.
(d) Check Optimization. If optimize then go to convergence Check otherwise loop is running until Objective form is not obtained.
Step 6: Check the convergence. If converge then check the cost features otherwise again initialize the particles and Repeat the step 5.
Step 7: If cost is less than Δ C then stop.

B. Proposed methodology: Flowchart - This section involves the proposed methodology using Genetic algorithm and Particle Swarm Optimization.
Flow chart 1

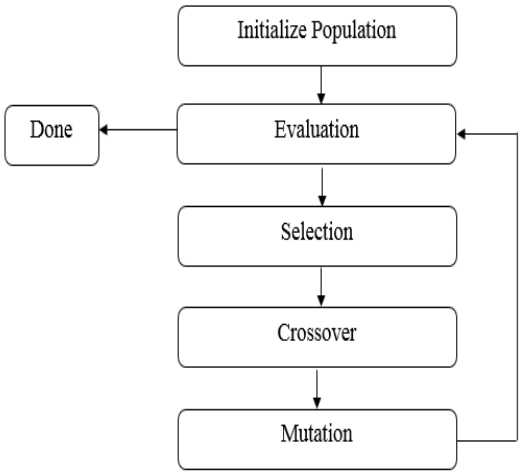

Figure 6: Flow chart of genetic algorithm

Flow chart 2

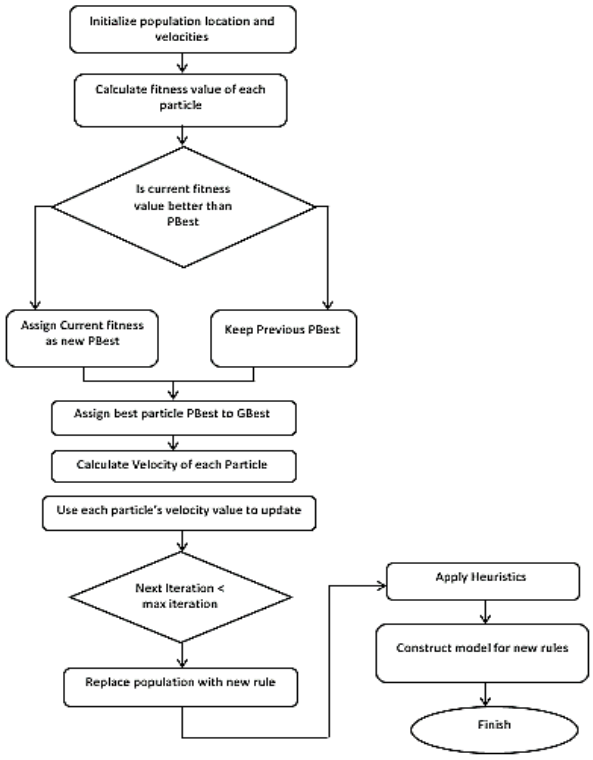

Figure 7: Flow Chart of PSO

Flowchart 3

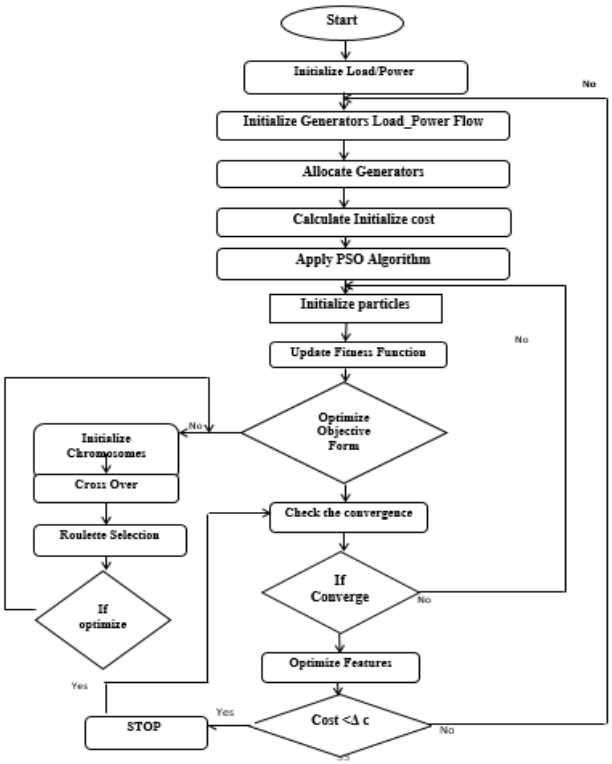

Figure 8: Flow Chart of PSO_GA

C. Proposed Algorithm -
i). Genetic algorithms: Genetic algorithm is a meta-heuristic algorithm which is used to solve the optimization problems in computing and artificial intelligence. It

provides the optimized solution by using the concept of selection and evolution. Genetic algorithms are able to solve the complex problems and provide reasonable solution on them. This algorithm is differing from the existing algorithms due to following reasons:-

- Genetic algorithm generates a population of point where as classical algorithm generates only single point in each iteration.
- Deterministic computation is used to select the next point in classical algorithms but genetic algorithm used random number generator to select the next population.

Genetic Algorithm works in the five stages that are following.

1. Initial Population: It is a set of individuals which is basically solutions of the given problem and called as population.

2. Fitness Function: In this phase fitness of each chromosome or solution is evaluated in the population set.

3. Selection: Two parent chromosomes are selected in this phase on the basis of their fitness.

4. Crossover: In this phase new population is created by this process called children. If this process is not occurred then offspring's are copy of parents.

5. Mutation: In this phase the mutation probability mutate new offspring at each position of chromosome.

Genetic Algorithm

Step 1: Population← initialize Population
Step 2: Evaluate the population.
Step 3: S_{Best}← get best solution from population.
Step 4: while (! Stop condition())
Parents ← select parents(Population, $Population_{Size}$)
Child← ∅
For($Parent_1 Parents_2 \in Parents$)
$Child_1 Child_2 \leftarrow Crossover\ (\ Parent_1 Parents_2 \in P_{Crossover}\ )$
$Children \leftarrow Mutate\ (Child_1, P_{mutation})$
$Children \leftarrow Mutate\ (Child_2, P_{mutation})$
End
Evaluate the Population of Children
$S_{Best} \leftarrow get\ best\ solution\ (Children)$
Population ← $replace\ the\ least\ fit\ population\ (children) with\ new$
End
Return (S_{Best})

ii). Particle Swarm Optimization: Optimizing the particle swarm may seem complicated, but it's really a very simple algorithm. On a number of iterations, a group of variables has its adjusted values closer to the member whose value is closest to the target at a given time. Imagine a flock of birds circling an area where they can smell a hidden food source. Whoever is closest to the food pips the loudest and the other birds sway in his direction. If one of the other birds gets closer to the target than the first one, he chirps harder and the others turn to him. This tightening pattern continues until one of the birds arrives on the food. It's a simple algorithm that is easy to implement.

The algorithm keeps track of three global variables:

1. Target value or condition
2. Best Global Value (gBest) indicating which particle data is currently closest to the target
3. Stop value indicating when the algorithm should stop if the target is not found

Each particle is composed of:

1. Data representing a possible solution
2. A velocity value indicating how much data can be changed
3. A better personal value (pBest) indicating the closest the particle's data has ever reached the target

PSO

Step 1: In PSO model for each particle i in S do
Step 2: for each dimension d in D do
Step 3: //initialize each particle's position and velocity
Step 4: $x_{i,d} = \boldsymbol{Rnd}(\boldsymbol{x_{max}}, \boldsymbol{x}_{\min})$
Step 5: $\boldsymbol{v_{i,d}} = \boldsymbol{Rnd}(-\boldsymbol{v_{max}}/3, \boldsymbol{v_{max}}/3)$
Step 6: end for
Step 7: //initialize particle's best position and velocity

$$\boldsymbol{v_i}(k+1) = \boldsymbol{v_i}(k) + \boldsymbol{\gamma 1_i}(\boldsymbol{p_i} - \boldsymbol{x_i}(k)) + \boldsymbol{\gamma_{2i}}(G - \boldsymbol{x_i}(k))$$

New velocity

$$\boldsymbol{x_i}(k+1) = \boldsymbol{x_i}(k) + \boldsymbol{v_i}(k+1)$$

Where
i- particle index
k- discrete time index
v_i –velocity of i^{th} particle
x_i – position of i^{th} particle
p_i- best position found by i^{th} particle(personal best)
G- best position found by swarm (global best, best of personal bests)
$G_{(1,2)i}$- random number on the interval[0,1]applied to the i^{th} particle
Step 8: $\boldsymbol{pb_i} = \boldsymbol{x_i}$
Step 9: // update global best position
Step10: if $\boldsymbol{f}(\boldsymbol{pb_i}) < \boldsymbol{f}(\boldsymbol{gb})$
Step 11: $\boldsymbol{gb} = \boldsymbol{pb_i}$
Step12: end if
Step13: end for

PSO_G.A

Step 1: Initialize the load/ power.
Step 2: Allocate the generators.
Step 3: Calculate the Initialize cost.
Step 4: In PSO model for each particle i in S do
Step 5: for each dimension d in D do
Step 6: //initialize each particle's position and velocity
Step 7: $x_{i,d} = \boldsymbol{Rnd}(\boldsymbol{x_{max}}, \boldsymbol{x}_{\min})$
Step 8: $\boldsymbol{v_{i,d}} = \boldsymbol{Rnd}(-\boldsymbol{v_{max}}/3, \boldsymbol{v_{max}}/3)$
Step 9: end for
Step 10: //initialize particle's best position and velocity

$$\boldsymbol{v_i}(k+1) = \boldsymbol{v_i}(k) + \boldsymbol{\gamma 1_i}(\boldsymbol{p_i} - \boldsymbol{x_i}(k)) + \boldsymbol{\gamma_{2i}}(G - \boldsymbol{x_i}(k))$$

New velocity

$$\boldsymbol{x_i}(k+1) = \boldsymbol{x_i}(k) + \boldsymbol{v_i}(k+1)$$

Where

i- particle index
k- discrete time index
v_i –velocity of i^{th} particle
x_i – position of i^{th} particle
p_i- best position found by i^{th} particle(personal best)
G- best position found by swarm (global best, best of personal bests)
$G_{(1,2)i}$- random number on the interval[0,1]applied to the i^{th} particle
Step 11: $\boldsymbol{pb_i}=\boldsymbol{x_i}$
Step 12: // update global best position
Step 13: if $\boldsymbol{f(pb_i) < f(gb)}$
Step 14: $\boldsymbol{gb = pb_i}$
Step 15: if the output is optimize then check the converge otherwise follow **Genetic algorithm for optimize results.**
Step 16: Population← initialize Population
Step 17: Evaluate the population.
Step 18: S_{Best}← get best solution from population.
Step 20: while (! Stop condition())
Parents ← select parents(Population, $Population_{Size}$)
Child← ∅
For($Parent_1 Parents_2 \in Parents$)
$Child_1 Child_2 \leftarrow Crossover\ (\ Parent_1 Parents_2 \in P_{Crossover}\ )$
$Children \leftarrow Mutate\ (Child_1, P_{mutation})$
$Children \leftarrow Mutate\ (Child_2, P_{mutation})$
End
Evaluate the Population of Children
$S_{Best} \leftarrow get\ best\ solution\ (Children)$
Population ← $replace\ the\ least\ fit\ population\ (children) with\ new$
End
Return (S_{Best})
Step 21: Check the convergence. If results are converged then optimize features are the output.
Step 22: Check the cost and stop.

IV. RESULT ANALYSIS

A. Comparative Results - In the section proposed result and comparison with different algorithm result is presented. This result is calculated on the heat generators and power generators on Genetic Algorithm, PSO, and Genetic with PSO.

Table.1 Overhead comparison values

Approaches	Overhead(100 iteration)	Overhead(200 iteration)	Overhead(500 iteration)
Genetic Algorithm	14.758	331.56	0.0321
PSO	125.48	69.0432	0.205
Genetic-PSO	0.0169	778.27	0.0253

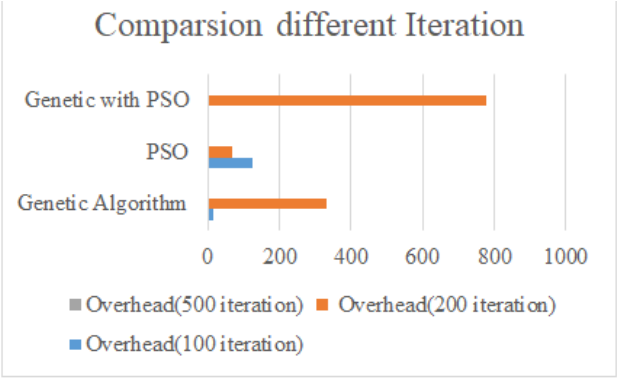

Figure 9: Overhead on different algorithms

Figure 9 depicts the heat generator values on the different algorithm approaches. The x-axis represents the algorithms and y-axis represents the values of the Overhead. The Genetic with PSO gives the effective heat generator values.

Table .2 Cost comparison values

Approaches	Cost (100days)	Cost (200days)	Cost (500days)
GA	49.11	79.7093	82.33
PSO	49.118	80	85.34
Genetic with PSO	49.11	74.34	80.34

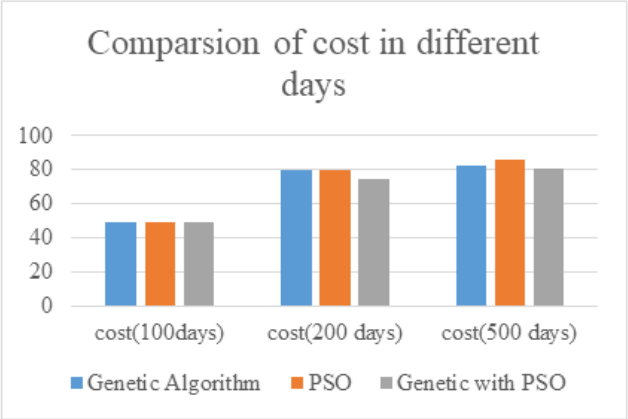

Figure 10: Power Generators on different algorithm

Figure 10 depicts the power generator values on the different algorithm approaches. The x-axis represents the algorithms and y-axis represents the values of the cost. The Genetic with PSO gives the effective power generator values. It shows the comparison of the heat and power generators on the different algorithms. Here x-axis shows the values and y-axis shows the cost in different number of days

Table 5.2 Results on different parameters.

Approaches	Number of order placed	overhead	Total cost(Rs)
Genetic Algorithm	345	1002.59	1727.68
PSO	500	472.74	1097.77
Genetic with PSO	789	320.13	835.33

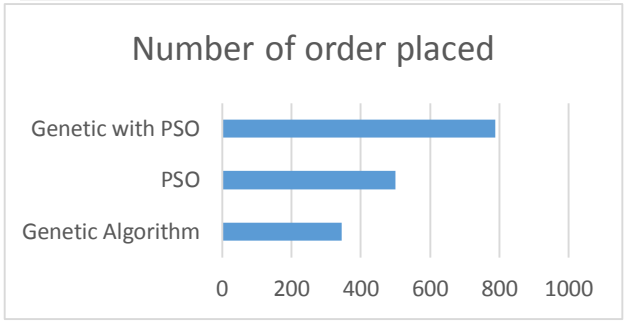

Figure 11: order placed on Different Algorithms

Figure 11 depicts the cost of heat in the different algorithms. In this graph x-axis represents the algorithmic approach and y-axis shows the value of order placed.

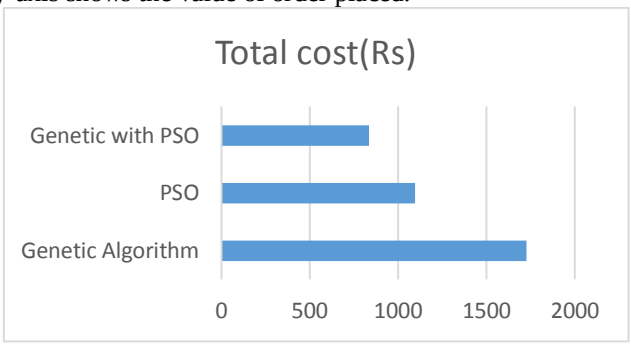

Figure 12: Cost on different algorithms

In figure 12, x-axis shows the approach used in the work and their comparison and y-axis represents the cost of the algorithms.

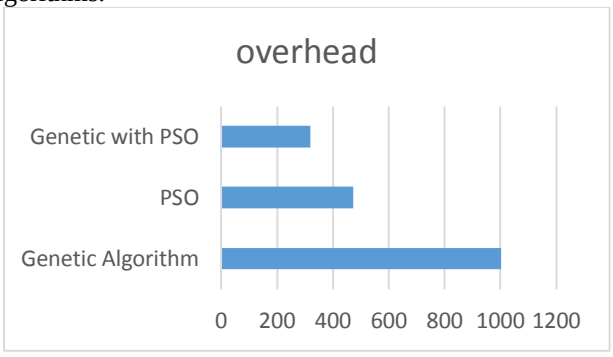

Figure 13: Total overhead on different algorithms.

Figure 13 represents the total cost of the algorithms which is represented by genetic algorithm, PSO and Genetic with PSO. The proposed hybrid approach Genetic with PSO represents the reduction in cost.

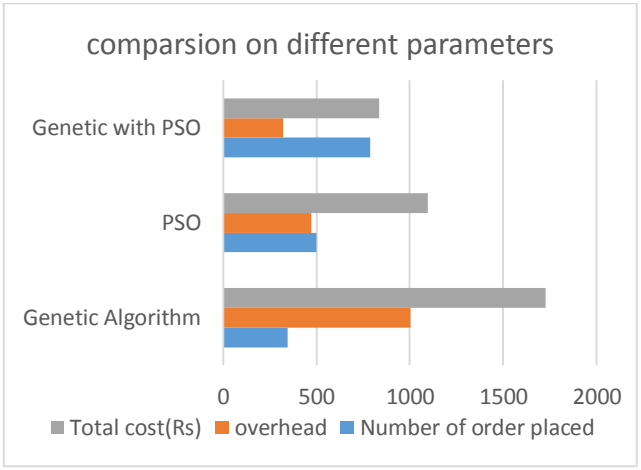

Figure 14: Cost Comparison on different algorithms

In figure 14, it depict the values of three algorithms that are Particle Swarm Optimization, G.A and Genetic Algorithm with Particle swarm optimization. The Blue bar represents the cost of Genetic algorithm, Red bar represents the Particle Swarm Optimization (PSO) and Green represents the proposed approach Genetic with PSO. The graph clearly describe the total cost is maximum on Genetic Algorithm and minimum on Genetic with PSO it is due to parallel working of both algorithm.

V. CONCLUSION

In this study, we dealt with the stochastic assembly line balancing problem for both the straight line and U-line configurations. Since the problem is known to be NP-hard, there are many heuristic methods developed for the assembly line balancing problem. This research proposed one such heuristic method for the stochastic assembly line balancing, with the objective of minimizing a line operating cost that consisted of both labor cost and task incompletion cost. We developed a hybrid optimization using genetic and particle swarm optimization based method for the cost oriented assembly line balancing problem. The idea, key to the core of this research is the concept of reduce cost and overhead but increase productivity. This idea removed the need for rule of thumb methods for closing a station by bringing a rationale behind this decision. The methodology for evaluating the expected cost of generated designs is taken from Kottas and Lau (1976). This procedure is exact and works by generating and enumerating all possible incompletion tuples. The solution methodology developed is tested on several problems of varying in size from 11 to 70 tasks. The test problems selected for use in this research are well known and well-studied problems. The solution found by the proposed method is compared to that of Kottas and Lau's (1981) algorithm. For the straight line balancing problem, the results obtained from experimentation on these problems reveals that the proposed heuristic. There are lines of research arising from this work which should be pursued: Firstly, probe the three algorithms chosen (Boctor,

Helgeson & Birnie and Bedworth & Bailey) in more examples with the aim to find out which is the best algorithm for that kind of problems. Finally, probe the new method with the post processor in bigger datasets to see if the new method performs well in real examples, and compare the results with the optimal ones.

VI. REFERENCES

[1]. Pereira, J., & Álvarez-Miranda, E. (2018). An exact approach for the robust assembly line balancing problem. *Omega*, *78*, 85-98.

[2]. Hazır, Ö, Delorme, X., & Dolgui, A. (2014). A survey on cost and profit oriented assembly line balancing. *IFAC Proceedings Volumes*, *47*(3), 6159-6167.

[3]. Kumar, N., & Mahto, D. (2013). Assembly line balancing: a review of developments and trends in approach to industrial application. *Global Journal of Research in Engineering*, *13(2)*, 2249-4596.

[4]. Baskak, M. (2008). Assembly Line Balancing in a Clothing Company. FIBRES & TEXTILES in Eastern Europe, 16(1), 66.

[5]. Haq, A. N., Rengarajan, K., & Jayaprakash, J. (2006). A hybrid genetic algorithm approach to mixed-model assembly line balancing. The International Journal of Advanced Manufacturing Technology, 28(3-4), 337-341.

[6]. Lapierre, S. D., & Ruiz, A. B. (2004). Balancing assembly lines: an industrial case study. *Journal of the Operational Research Society*, *55*(6), 589-597.

[7]. Karabatı, S., & Sayın, S. (2003). Assembly line balancing in a mixed-model sequencing environment with synchronous transfers. *European Journal of Operational Research*, *149*(2), 417-429.

[8]. Becker, C. and Scholl, A. (2003). "A survey on problems and methods in generalized assembly line balancing", European Journal of Operational Research ISSN: 1611-1311.

[9]. Lee, T. O., Kim, Y., & Kim, Y. K. (2001). Two-sided assembly line balancing to maximize work relatedness and slackness. *Computers & Industrial Engineering*, *40*(3), 273-292.

[10]. Carnahan, B. J., Norman, B. A., & Redfern, M. S. (2001). Incorporating physical demand criteria into assembly line balancing. *IIE Transactions*, *33*(10), 875-887.

[11]. Merengo, C., Nava, F., & Pozzetti, A. (1999). Balancing and sequencing manual mixed-model assembly lines. *International Journal of Production Research*, *37*(12), 2835-2860.

[12]. Baybars, I. (1986). A survey of exact algorithms for the simple assembly line balancing problem. *Management science*, *32*(8), 909-932.

[13]. Salveson, M. E. (1955). The assembly line balancing problem. The Journal of Industrial Engineering, 18-25.

[14]. Assembly Systems and Line Balancing. Available at: http://www.me.nchu.edu.tw/lab/CIM/www/courses/Flexible%20Manufacturing%20Systems/Microsoft%20Word%20%20Chapter8FbASSEMBLY%20SYSTEMS%20AND%20LINE%20BALANCING.pdf assessed on 02/04/2019 at 6.00 PM.