Efficient Application of Metaheuristic Algorithms for Balancing Human Robot Collaborative Assembly Lines

A REVIEW 2 PROJECT REPORT

Submitted in partial fulfillment for the award of the degree of

B.Tech

in

Mechanical Engineering

by

Ranganathan S V – 18BME0374

Sridharan A P – 18BME0510

School of Mechanical Engineering

FEBRUARY & 2022

TABLE OF CONTENTS

ABSTRACT

In the wake of industry 5.0 practices, resilient production technologies are advancing progressively due to increasing emphasis in shifting towards sustainability. Hence, the global demand curve is constantly on the rise, owing to technological and efficiency advancements in every product line. To follow up on this constant utility development, industries find it vital to adapt and capture the shifting market value at a rapid phase. The project aims to balance and improve the efficiency of different assembly line tasks that involve collaborative robots working hand in hand with humans. This is to be done through development of optimization algorithms by varying current metaheuristic algorithms, with an intention of less complex data processing and increased efficiency. Parallely, during the fine tuning of the developed algorithm, multi-objectives (cycle-time reduction, ergonomic & safety enhancement, energy consumption reduction) are considered along with real life assembly line constraints. Moreover, the developed algorithms are to be consecutively compared with several benchmark functions and datasets and their relative performance evaluated.

CHAPTER 1

INTRODUCTION

Assembly line in the manufacturing process comes at last which delivers the finished product with a sequence of several tasks. Normally these tasks were performed by humans earlier before the introduction of robots. However, due to ever-changing consumer tastes and to outperform the competitors, many industries started using robots in their assembly line. The main reasons for this change are to improve production rate, efficiency, reduce cost(wastes) and to increase agility towards changing demands.

This use of robots in assembly lines was called as **robotic assembly lines(RAL)**. Use of robots in industries helped them to reduce labour cost, variability in production, and enhance productivity as robots can work 24/7 all year with no fatigue or illness.

Advancements in technology helped this evolution to further replace humans by robots in performing complex tasks too. Since the introduction of robots in assembly lines, **robotic assembly line balancing(RALB)**, and extension of ALB have become of interest for many researchers and industries to further optimize their production. This varied from the conventional ALBP as here two different problems are to be addressed and they are, assignment of tasks to the given workstations and assignment of different available robots to the workstations.

The general assembly line balancing problem has some objective such as minimising cycle time, cost, etc.., with some constraints such as precedence relation of tasks, number of workstations, etc.., As the robots are accurate in performing tasks, their task times are assumed to be deterministic while solving RAL BP. Some other assumptions made by the researchers on solving them were,

- The precedence relations of tasks are known and the task times depend completely on the type of robot chosen.
- One workstation comprises only one robot and all types of robot are always ready to use without any capacity and cost limitations.

Further RALBP is classified into several types such as,

- ➔ RALBP TYPE I – Aims at reducing the number of workstations by assigning best suited robots and tasks to workstations.
- ➔ RALBP TYPE II – Aims at minimising cycle time with a predefined number of workstations.
- ➔ RALBP TYPE E – Aims at reducing both cycle time and workstations, thus maximising the assembly line efficiency.

- ➔ RALBP TYPE F – Aims at finding feasible solutions for a predefined set of workstations and cycle times.
- ➔ RAL BP TYPE COST – Aims at the monetary and economic aspects of RAL.
- ➔ RAL BP TYPE O – Other categories are classified in this type.

In RALB, the classifications are based on the structure of the assembly lines**[AL - assembly line]**. They are MAL(multi manned), PWAL(parallel workstation), PAL(parallel), UAL(U-shaped), StAL(straight line), 2SAL(two sided), PUL(parallel U-line), PAUL(parallel adjacent U-line), PMAL(parallel multi-manned).

As the automation in the manufacturing department is increasing significantly, several robots are developed to carry out several tasks to increase the production rate. However, not all tasks can be automated due to the lack of flexibility with available robotic technologies. Humans on the other hand are flexible, adaptable according to market demands, and also have decision making skills with creativity. To remain relevant, many manufacturing enterprises show interest in human robot collaboration where the skills of humans and the speed, endurance, and accuracy of robots are brought together to increase production. This also helps the human workers as the repetitive and stressful tasks are done by robots.

Many algorithms and meta-heuristics were used to find the best allocation of robots and humans to workstations, determine the optimal task sequence, cycle time, etc.., Some are,

1) Artificial Bee Colony Algorithm (ABC)
2) Genetic Algorithm (GA)
3) Differential Evolution (DE)
4) Teaching Learning based optimization (TLBO)
5) Particle swarm optimization (PSO), etc..,

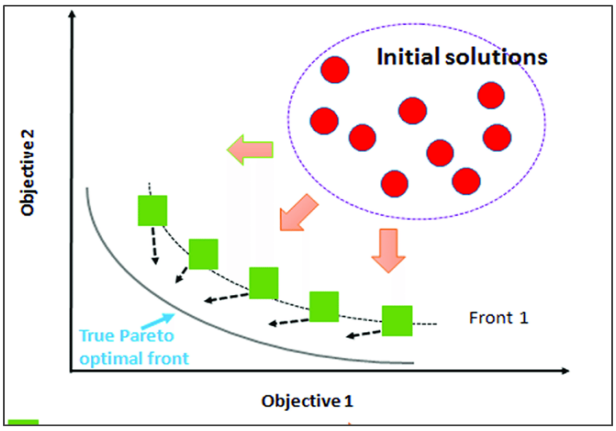
Multi objective optimisation refers to solving multiple objectives that are conflicting in nature. As the objectives are conflicting, rather than a single optimal solution, a set of optimal solution is presented known as the pareto optimal set. In non-dominated sorting, an individual A is said to dominate another individual B, if and only if there is no objective of A worse than objective of B. In simple words, let us say there are two objectives f1 & f2 that is to be minimised, and two points A and B,

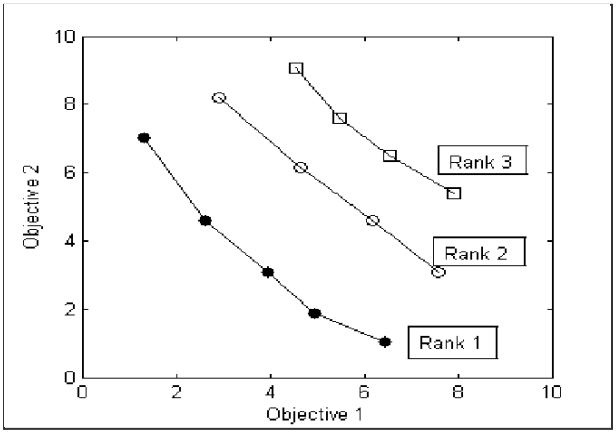
- **A dominates B if**, $[f1(A) < f1(B)]$ **&** $[f2(A) < f2(B)]$
- **Both lie in same front if**, $[f1(A) < f1(B)]$ **&** $[f2(A) > f2(B)]$

or [f1(A) > f1(B)] **and** [f2(A) < f2(B)]

- **B dominates A if**, [f1(A) > f1(B)] **and** [f2(A) > f2(B)]

Some of the objectives in assembly line balancing problems are, To minimize the number of workstations, To minimize the cycle time, To minimize the energy consumption, To maximize the workload smoothness, etc..,





Some of the classic techniques used in handling multi-objective decisioning are listed below along with their disadvantages.

WEIGHTED SUM METHOD - Constructs a weighted sum of objectives and optimizes them. The difficulty with this method is that, the user must be well knowledged with the weights for each objectives as it influences the solution obtained.

CONSTRAINT METHOD - Minimizes a primary objective while expressing all the other objectives in the form of inequality constraints. The difficulty with this method is that if the objectives are complex, expressing them into constraints is not effective in generating pareto optimal solutions.

NON DOMINATED SORTING WITH EXTERNAL ARCHIVE - This method as explained previously, compares and simultaneously generates many different solutions and stores them separately in an external archive. This external archive at the termination of iterations yields the best multi-objective optimum solution. This method of NSGA - II has the most widespread use in handling multi-objective decisioning due to faster computation, reduced complexity and accepted method of approach.

CHAPTER 2

LITERATURE REVIEW

[1] Puts forth three different layouts for human-robot assembly line (only human/both in a workstation/both in different workstations). An MILP approach to solve every scenario was formulated, which was referred for our problem synthesis.

[2] Tabulates all possible risks in a human-robot collaborative assembly line environment, along with precedence insights of every risk. Risks are referred to effectively formulate our problem in objective of increasing ergonomics.

[3] RALB problem solution approaches and considerations that weren't accounted for previously.

[4] Insights of best performing algorithms are compared in their efficient performance, high number of citations, specific evolutionary operators, interesting interaction mechanisms between individuals, parameter tuning/handling concepts, and stagnation prevention methods.

[5] In addition to the traditional ALBP constraints, this paper is referred for human and robot characteristics in terms of task times, allowing multiple humans and robots at stations, and their joint/collaborative task formulations. Also, how real time industrial case study is carried out.

[6] A framework that represents the complexity levels of the influencing factors in H-R workplace. The guidelines and recommendations from literature are categorised into 3 groups of H-R design layout and considerations, with increasing order of complexity.

[7] Presents the cost-oriented robotic assembly line balancing problem with setup times to minimize the cycle time and total purchasing cost simultaneously. The NSGA-II is developed for multi-objective implementation and improved phases in ABC algorithm are developed to achieve a set of Pareto solutions.

[8] An analytical method is introduced to evaluate the flow time of an assembly process with collaborative robots. The productivity performance of such systems are measured and system properties of monotonicity, work allocation, and bottlenecks are investigated. Also, real time industrial case study is carried out.

[9] This study investigates the mixed-model ALBP with the collaboration between human workers and robots. Bee and ABC algorithms are formulated with new phases each to increase exploration and exploitation. ABC proposes a new onlooker phase to accelerate the evolution of

swarm, new scout phase to achieve high-quality solutions and preserve diversity of swarm, and local search to enhance exploitation.

[10] The complete methodology of MBO algorithm applied for assembly line balancing with collaborative robots. Provides insights on hybridisation of MBO and use of NSGA-II for solving multi-objective problems.

[11] Shows the implementation methodology of AOA algorithm, its meaning and significance for solving optimization problems.

CHAPTER 3

GAPS IN THE LITERATURE

- Since the study started on RAL BP, only 33 articles have been published to date since 1991(over 30 years). The research on this field was less and skyrocketed recently (2015-2019).
- In 1997, most of the researchers started using deterministic variable task times that were determined by the types of robots. This was due to the development in technology that resulted in the invention of many different robots that can perform all the tasks.
- Mathematical models were used to describe the relations between the different objectives and to handle the constraints effectively. As the size of the problem increased, meta-heuristics started dominating the other methods. Many meta-heuristics were developed to tackle this NP-hardness of the RALBP.
- Introduction of Cobots(collaborative robots) to the assembly line created a new trend in the research area of RALBP. This use of cobots in the assembly line showed better results and gained interest by many industries. **As this area of research is quite new, only a few papers have been published since 2019.**
- Many real life constraints such as tools assignment, tool space optimization, etc.., were not addressed in most of the papers and this could be the future in the study of RAL BP.
- Almost all the research on RALBP assumes a fully automated assembly line which requires huge investments. Not all the industries are capable of making such investments

and still many of the industries slowly started to introduce robots in their assembly lines. So the **study of human robot collaboration assembly lines could be a great research area as many industries opt for them as they are easy and require less investment.**

- The studies that included cobots in the assembly line mostly preferred only one robot in a workstation. Research study on multiple robots and humans working in parallel could be a great area of research. As humans started working alongside robots, study of ergonomics and real life constraints such as human fatigue, sequence dependent setup times, tool and accessory changing, part transportation between stations, etc.., can be included in RABLP as no one has ever done that previously due to its high complexity.

- Most industries focus on only one objective while designing assembly lines rather than many. **Research on multi-objective optimizations could be more relevant to real life problems.** As the introduction of multi-objectives created more complexity, many meta-heuristics were developed. From the recent studies, hybrid metaheuristics showed better results than meta heuristics. So study on hybrid metaheuristics on solving multiple objectives could create a way for finding better pareto optimal solution sets.

- In-line with the growing interests and developments in human-robot collaborative assembly line balancing problems using metaheuristics, **"standard testing datasets" of different task sizes for H-R ALBP aren't available for researchers who may use such reliable and ready-made datasets to test different algorithms.**

- Due to several intricate considerations in designing a H-R collaborative assembly line, **a broad range of publications in this little researched domain haven't "collectively" emphasised flexible and realistic constraints with respect to individual task abilities of humans and robots, collaborative work safety, economic production.**

- A very recent algorithm called "ARCHIMEDES OPTIMISATION ALGORITHM" or AOA is implemented to solve our H-R ALB problem. **This algorithm is very recent, has less development fatigue and importantly, has never been used for solving any line balancing problems before. To make things interesting, this algorithm performs slightly better than all algorithms considered by us for our problem.**

CHAPTER 4

PROBLEM DEFINITION

The project aims to balance and improve the efficiency of different assembly line tasks that involves collaborative robots working hand in hand with humans. This is to be done through development of optimization algorithms by varying current metaheuristic algorithms, with an intention of less complex data processing and increased efficiency.

PROBLEM STATEMENT:

- To develop metaheuristic algorithms for balancing and scheduling of assembly line operations.
- Consider assembly line structure = StAL (straight line assembly line).
- Only one product is assembled in the assembly line with a known number of workstations.
- Considering setup times and sequence dependent time.
- Collectively considering constraints of ergonomics, safety, economic viability of assembly line resources.

This problem type is classified as a multi **objective RAL BP TYPE - II** that focuses on *minimising cycle time and energy consumption with a given number of workstations.*

CHAPTER 5

OBJECTIVES

- To develop metaheuristic algorithms for balancing and scheduling of Human-robot collaborative assembly line operations.
- Considering the assembly line structure of StAL (straight line assembly line) where a single product is assembled with a known number of workstations, to use the developed metaheuristic algorithms to optimize multiple objectives of CYCLE TIME REDUCTION, REDUCING OVERALL LINE ENERGY CONSUMPTION, with an overall IMPROVEMENT IN ERGONOMIC, ECONOMIC & LAYOUT CONSTRAINTS.
- To consider setup times, sequence dependent times and other real-time line constraints

that haven't been considered to solve human-robot ALB problems so far.

CHAPTER 6

WORK CARRIED OUT SO FAR

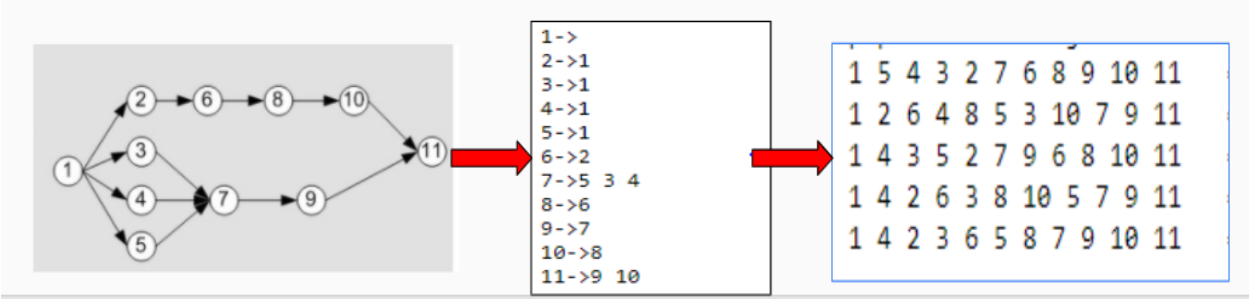
Literature Survey - We conducted an extensive literature survey and studied various assembly line balancing techniques of only workforce, robotic lines as well as human-robot lines. Parallely, metaheuristic algorithms and their benchmark function efficiencies were reviewed to decide the best algorithm to proceed. We have finalized to code our own HYBRID algorithm of PSO, TLBO, MBO and AOA that are suited to our problem environment and considerations.

6.1 SEQUENCE GENERATION:

For generating sequences, a precedence matrix is generally used in many papers.
But in our proposed method, we use an adjacency list that stores the precedence relation efficiently and helps in generation of sequences that satisfy these relations.
The algorithm is as:

- First an available set is created with the tasks that have no precedence tasks.
- Then one task is selected randomly from that set and the tasks following it are added to the available set.
- This step is repeated until all the tasks are assigned and the available set is empty.

The use of adjacency lists is very helpful in implementing this algorithm.
Adjacency holds the data as shown in this example.



6.2 OBJECTIVE FUNCTION EVALUATION (cycle time):

For calculating the cycle time of a particular sequence generated, a consecutive method algorithm is used. The working of this algorithm is:

- Cycle time is initially fixed with a value by summing up the minimum task time of all robots.
- For each station, the sequence is tried with all the available robots with a given number of stations and the cycle time that was found previously.

- In each iteration, if the found cycle time is not feasible, the cycle time is incremented.
- This step is repeated until a feasible cycle time is found which is the optimal cycle time for the given sequence.

For example, the image on the right shows an example of allocation of tasks in each workstation using the consecutive method.
As we can see here, the overall cycle time of such allocation is maximum cycle time, i.e.., **202** for the sequence **[1, 5, 3, 4, 7, 9, 2, 6, 8, 10, 11]** with **4 workstations.**

```
station number 1's allocation:
robot assigned : 3
tasks assigned : 1 5 3
its cycle time : 201

station number 2's allocation:
robot assigned : 4
tasks assigned : 4 7 9
its cycle time : 202

station number 3's allocation:
robot assigned : 4
tasks assigned : 2 6
its cycle time : 152

station number 4's allocation:
robot assigned : 2
tasks assigned : 8 10 11
its cycle time : 202
```

6.3 FORMULATION OF OUR HYBRID ALGORITHMS OF PSO, TLBO, MBO & AOA:

For implementing the standard algorithms for a discrete problem, some assumptions and formulations are made such as,

- ❖ For encoding purposes, only the sequence array is used.
- ❖ For decoding purposes, the robots assigned and the tasks performed in each station are determined by knowing the optimal cycle time.
- ❖ The addition of setup times and sequence dependent time is used to make the problem more suitable for industrial purposes.
- ❖ Setup time is added with the task time for each task performed by the robot.

- Sequence-dependent time is added only for the tasks that are not first in the station as these times are dependent on the previously performed tasks(the first performed task in each station by the robot has no previous task).
- Variants for algorithms are created to make the algorithm give better results.
- By hybridizing the algorithm with another combines the advantages of both the algorithm and gives a better solution than a single algorithm.
- The algorithm chosen to hybridize with the developed 4 algorithms is ABC (artificial bee colony algorithm).
- The scout phase from this ABC algorithm is added with those 4 algorithms(PSO, TLBO, MBO, AOA).
- The main reason for adding the scout phase is that it increases the exploration and exploitation space and gives better pareto fronts by replacing the retained solutions and replacing it with a random one.
- Easy to implement and hybridize with other algorithms.
- All these algorithms are compared using those performance indicators. The best performing algorithm after testing those with different data sets is presented in the results & discussion section.

```
/* Initialization */
Site initial food sources using (1)
Cycle = 1
For each food source(i)
   unsuccessfulTrials(i) = 0
End
Repeat
  /* Employed Bees Operator*/
  For each employed bee(i)
      Send employed bee to corresponding food source(i)
      Site a new food source using (2)
      Apply greedy selection
      If new food source selected
         unsuccessfulTrials(i) = 0
       Else
         unsuccessfulTrials(i) = unsuccessfulTrials(i) + 1
  End
  /* Onlooker Bees Operator */
  Calculate probability P_i for each food source(i) using (3)
  For each food source(i)
      Send onlooker bees to food source(i) in accordance
      with associated P_i
      For each onlooker bee(j)
         Site a new food source using (2)
         Apply greedy selection
         If a new food source selected
            unsuccessfulTrials(i) = 0
         Else
            unsuccessfulTrials(i) = unsuccessfulTrials(i) + 1
       End
  End
  /* Scout Bee Operator */
  For each food source(i)
      If  unsuccessfulTrials(i) > limit
         Abandon the food source(i)
         Site a new food source using (1)
  End
  Memorize the best solution achieved so far
  Cycle = Cycle + 1
Until termination criteria are met
```

6.4 DATA GENERATION:

Data required for our problem are as follows:

- ➔ Number of tasks.
- ➔ Number of robots.
- ➔ Operation time table for all robots, cobots and humans.
- ➔ Setup times for robots.
- ➔ Sequence dependent times for robots.
- ➔ Precedence matrix.
- ➔ Energy Consumption by robots and humans.

Some of the assumptions made while generating the data are,

- Number of cobots = Number of robots
- Cobot_i = Robot_i + Human
- Only one type of worker (all workers are equally skilled)
- The lower bound and upper bound of operation times are used for calculation of lower and upper bounds of other data (setup time, sequence dependent time).
- Energy consumption of robots and humans is given in Wh/min. **[0.10 - 0.30]** Wh / min

The parameters to be included in the H-R ALBP data are as follows:

- Number of tasks.
- Number of robots.
- Lower bound(**lb**) and upper bound(**ub**) for operation time of robots table.
- **c1, c2, c3, c4** (parameters [**range from 0 to 1**] that control the lower and upper bounds of setup time and sequence dependent time).
    - ➔ Lower bound of operation time for cobot = lb
    - ➔ Upper bound of operation time for cobot = ub - (c1 * (ub - lb))
    - ➔ Lower bound of operation time of human = lb + (c2 * (ub - lb))
    - ➔ Upper bound of operation time of human = ub + (c2 * (ub - lb))
    - ➔ Lower bound of sequence dependent time = c3 * lb
    - ➔ Upper bound of sequence dependent time = c4 * lb
    - ➔ Lower bound of setup time = c1 * lb
    - ➔ Upper bound of setup time = c2 * lb

MATRIX REPRESENTATION OF DIFFERENT INPUT DATA FORMATS IN OUR PROBLEM:

- **OPERATION TIMES:**
  Is assumed to be in minutes and the data type used is integer.
  Dimension = r X n, where
  r = (number of robots * 2) + 1
  n = number of tasks

OPERATION TIMES	Task 1	Task 2	Task 3	Task 4
ROBOT 1				
ROBOT 2				
ROBOT 3				
ROBOT 4				
COBOT 1 (ROBOT1 + HUMAN)				
COBOT 2 (ROBOT2 + HUMAN)				
COBOT 3 (ROBOT3 + HUMAN)				
COBOT 4 (ROBOT4 + HUMAN)				
HUMAN				

- **SETUP TIMES:**

  Is assumed to be in minutes and the data type used is integer.

  Dimension = r X n, where,

  r = number of robots

  n = number of tasks

SETUP TIMES	Task 1	Task 2	Task 3	Task 4
ROBOT 1				
ROBOT 2				
ROBOT 3				
ROBOT 4				

- **SEQUENCE DEPENDENT TIMES:**

  Is assumed to be in minutes and the data type used is integer.

  Dimension = r X [n X n], where
  r = number of robots
  n = number of tasks

SEQUENCE DEP TIMES FOR ROBOT - i	Task 1	Task 2	Task 3	Task 4
Task 1				
Task 2				
Task 3				
Task 4				

- **PRECEDENCE MATRIX:**

The data in this matrix is either 0 or 1 where 0 means no precedence relation and vice versa. Dimension = [n X n], where n = number of tasks.

PRECEDENCE MATRIX	Task 1	Task 2	Task 3	Task 4
Task 1				
Task 2				
Task 3				
Task 4				

- **ENERGY CONSUMPTION:**

  The data is in the form or array where the energy consumption per min of robots and humans are given. Data type used is double.
  Dimension = [1 X (r+1)], where r = number of robots

ENERGY CONSUMPTION	ROBOT 1	ROBOT 2	ROBOT 3	ROBOT 4	HUMAN
IN [Wh / MIN]					

SAMPLE RESULTS OBTAINED FROM INPUT CONSIDERATIONS:

```
c1 = 0.25; //tuning lower bound for setup time
c2 = 0.5; //tuning upper bound for setup time
c3 = 0.1; //tuning lower bound for seq dep time
c4 = 0.25; //tuning upper bound for seq dep time
//cout<<endl<<"ENTER NUMBER OF TASKS : ";
no_of_tasks = 11;
//cout<<endl<<"ENTER NUMBER OF ROBOTS : ";
no_of_robots = 4;
//cout<<endl<<"ENTER LOWER BOUND FOR OPERATION TIME(minutes)(int)  : ";
lb = 50;
//cout<<endl<<"ENTER UPPER BOUND FOR OPERATION TIME(minutes)(int) : ";
ub = 100;
```

```
NUMBER OF TASKS : 11
NUMBER OF ROBOTS : 4
LOWER BOUND FOR OPERATION TIME : 50
UPPER BOUND FOR OPERATION TIME : 100
LOWER BOUND FOR SETUP TIME : 12
UPPER BOUND FOR SETUP TIME : 25
LOWER BOUND FOR SEQUNCE DEPENDENT TIME : 5
UPPER BOUND FOR SEQUENCE DEPENDENT TIME : 12
```

```
SEQUENCE DEPENDENT TIME FOR ROBOT 1
9 7 9 9 7 6 11 6 11 10 5
12 6 12 8 8 5 9 9 11 11 8
7 10 8 10 7 11 12 11 7 9 5
11 5 7 5 11 8 11 8 9 11 10
8 6 5 8 10 9 6 8 12 8 6
8 6 8 6 5 6 8 9 6 7 10
9 7 8 12 5 12 8 11 9 11 12
9 7 9 6 8 5 5 12 6 8 5
9 10 5 10 5 10 11 7 7 7 9
10 7 10 9 10 8 5 9 8 10 11
12 11 6 12 11 5 5 7 5 9 12

SEQUENCE DEPENDENT TIME FOR ROBOT 2
6 6 12 11 5 7 5 7 11 10 9
8 7 7 12 7 11 7 12 9 6 10
10 6 9 11 6 11 11 11 10 12 12
9 10 12 11 10 7 10 8 11 5 10
5 12 12 11 6 12 7 8 9 5 9
5 11 10 11 9 8 8 9 8 5 6
7 11 12 9 8 7 8 9 12 8 8
11 7 10 10 9 5 7 9 9 7 7
6 6 12 10 9 8 5 9 9 7 8
8 12 11 10 7 7 9 10 11 8 12
8 5 9 8 7 5 12 10 8 5 11

SEQUENCE DEPENDENT TIME FOR ROBOT 3
7 10 7 10 10 12 6 5 7 10 12
5 7 6 8 12 11 6 7 11 9 7
7 12 10 7 11 7 10 11 5 12 9
7 9 6 6 11 6 8 8 5 9 10
6 12 9 5 5 11 11 9 6 5 8
11 7 6 5 5 12 5 12 8 7 9
10 9 7 11 12 10 12 8 7 5 7
12 5 7 10 11 11 11 11 6 9 6
7 9 6 7 9 5 10 12 9 7 8
11 6 7 8 5 11 11 5 5 10 6
8 7 12 6 6 11 8 10 12 10 7

SEQUENCE DEPENDENT TIME FOR ROBOT 4
5 12 11 5 10 10 10 12 5 8 5
8 12 5 6 10 6 6 7 7 9 9
6 11 10 12 6 8 11 11 10 11 11
8 12 8 6 9 7 6 12 8 9 11
8 10 8 9 12 10 11 8 7 5 6
12 12 7 7 11 6 12 9 12 8 8
7 9 12 9 10 12 12 7 10 8 12
6 12 11 11 11 7 5 11 8 5 10
11 7 8 12 7 5 11 10 8 5 6
8 9 11 7 9 5 12 12 5 5 11
11 12 9 5 12 7 9 12 5 7 7
```

```
PRECEDENCE MATRIX
0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0 0 0
1 1 1 0 0 0 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0
1 0 0 1 1 0 0 0 0 0 0
1 0 1 0 1 1 0 0 0 0 0
1 1 0 0 0 0 0 0 0 0 0
0 0 0 1 1 0 1 0 0 0 0
1 0 1 1 0 0 1 0 0 0 0
0 1 0 1 1 1 1 0 1 1 0
```

```
SETUP TIMES FOR ROBOTS
22 14 19 14 18 18 23 23 25 20 22
13 18 12 20 23 14 25 25 23 13 23
15 21 21 20 22 25 22 15 15 18 17
22 19 24 13 14 21 12 23 16 12 15
```

```
ENERGY CONSUMPTION
0.25 0.21 0.11 0.2 0.13
```

```
OPERATION TIMES FOR HRC
94 85 63 94 68 56 70 58 91 100 51
90 76 72 56 92 94 75 74 66 61 76
54 53 58 89 60 58 69 75 90 87 84
52 54 51 58 99 85 74 98 86 63 98
58 58 64 78 87 71 75 61 76 69 54
65 74 78 74 66 65 76 60 54 80 74
84 69 75 51 86 50 72 67 71 59 75
63 87 52 84 52 63 72 87 67 66 73
98 116 124 88 82 111 88 101 102 88 103
```

SAMPLE ENCODED AND DECODED TASK ALLOCATION BASED ON INPUTS:

→ A sequence of tasks (following precedence relation) is used to represent the order of tasks to be performed. Similarly, a sequence of process alternatives representing numbers between 1-3 for each station representing,

- 1 - Only robot is allocated
- 2 - COBOT (robot + human) is allocated
- 3 - Only human is allocated

For example, a **25 tasks, 6 stations** encoding sequences looks like,

TASK PERMUTATION - 1 2 3 4 8 9 5 6 10 7 12 15 11 13 14 20 17 21 16 18 24 23 25 19 22

PROCESS ALTERNATIVES - 3 1 2 1 2 1

```
1 2 3 4 8 9 5 6 10 7 12 15 11 13 14 20 17 21 16 18 24 23 25 19 22
3 1 2 1 2 1

station number 1's allocation:
human assigned
tasks assigned : 1 2
its cycle time : 390

station number 2's allocation:
robot assigned : 1
tasks assigned : 3 4 8 9
its cycle time : 483

station number 3's allocation:
cobot assigned : 1
tasks assigned : 5 6 10 7
its cycle time : 419

station number 4's allocation:
robot assigned : 1
tasks assigned : 12 15 11 13 14
its cycle time : 513

station number 5's allocation:
cobot assigned : 3
tasks assigned : 20 17 21 16 18
its cycle time : 491

station number 6's allocation:
robot assigned : 2
tasks assigned : 24 23 25 19 22
its cycle time : 516
```

FORMULATION OF HUMAN ROBOT COLLABORATION IN ASSEMBLY LINE BALANCING FOR OUR PROBLEM:

- Not all tasks can be automated due to the lack of flexibility with available robotic technologies. Humans are *flexible, adaptable according to market demands, and also have decision making skills with creativity.* Therefore, human robot collaboration is prevalent where the skills of humans and the speed, endurance, and accuracy of robots are brought together to increase production. This introduction of human robot collaboration leaves us with three possibilities, to find the best possible assembly line task allocation combination.

    1. Only workers are present in workstation
    2. Only robots are present in the workstation
    3. Both workers and robots are present in the workstation

- **TASK CONSTRAINTS** with respect to the ability of each type of robot, cobot(human+worker) and human in carrying out each individual task are taken into account. Inculcation of task constraints proves flexible and realistic for any entity that needs a fix to their semi-automated assembly line. In our problem, we take this into account by defining a constraint matrix that defines the "task ability/disability" (represented by 1 & 0 respectively) of all possible combinations of the 3 aforementioned workstation configuration possibilities with each listed task.

CONSTRAINT	TASK 1	TASK 2	TASK 3	TASK 4	TASK 5
ROBOT 1	1	1	0	1	1
ROBOT 2	1	1	1	1	1
ROBOT 3	0	0	1	1	0
COBOT 1 (robot 1 + human)	1	1	1	1	1
COBOT 2 (robot 1 + human)	1	0	1	1	1
COBOT 3 (robot 1 + human)	1	1	1	0	0
HUMAN	0	1	0	1	0

➢ **WORKSTATION CONSTRAINT** is another such realistic model inculcated in line with other constraints. The constraint is such that "no two consecutive workstations shall have the same worker configuration". This constraint adds to the practicality of limited automation investments in the line, as well as a spread in workforces, leveraging the flexibility and utter ability of humans to tackle problems at any point of the assembly line.

CHAPTER 7

METHODOLOGY

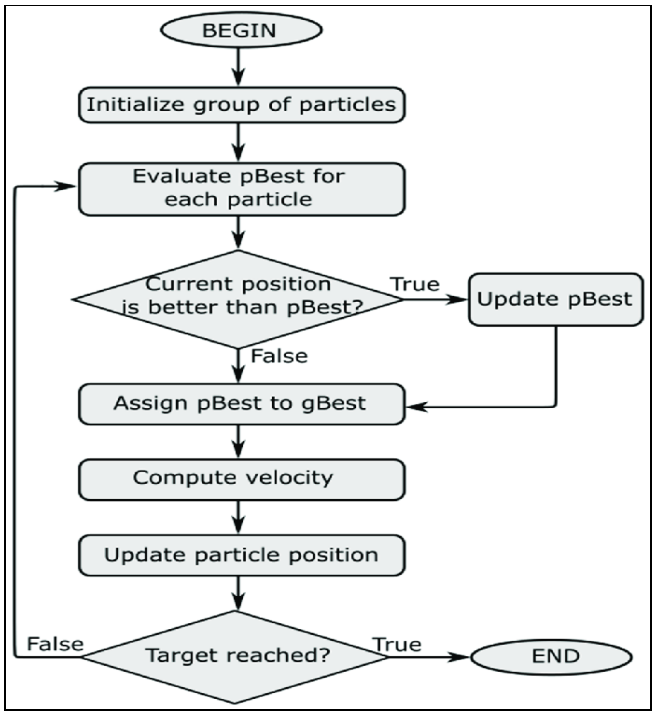
7.1 WORKING OF MODIFIED DPSO:

$$X_i^{t+1} = X_i^t + V_i^{t+1}$$

$$V_i^{t+1} = c_1 V_i^t + c_2(Pbest_i^t - X_i^t) + c_3(Gbest_t - X_i^t)$$

NOTATIONS USED:

t = indicates the iteration number
Xi = a single i'th solution in the population
Vi = a single i'th velocity vector in the population
P_best_i = indicates the best solution obtained by i'th particle so far
G_best = indicates the best solution obtained by the whole swarm so far
c1 = inertia weight
c2 = acceleration coefficient of P_best
c3 = acceleration coefficient of G_best



POPULATION UPDATION:

For implementing the modified DPSO method, the velocity vector is taken as an array of integers ranging from [0 - no.of.tasks] instead of velocity pairs that are used in most of the papers. These velocity vectors are used to update the population using certain conditions. They are,

SUBTRACTION (position - position)

Let X1,t = [x1,1, x1,2, x1,3, x1,4, x1,5, x1,6, x1,7]
X2,t = [x2,1, x2,2, x2,3, x2,4, x2,5, x2,6, x2,7]
V1,t =X1,t - X2,t
In this case, if x1 and x2 in the jth position are equal, then v1= 0. Otherwise, v1=x1. For example,

```
X1,t = 2 3 1 4 5
X2,t = 1 3 5 4 2
V1,t = X1,t - X2,t = 2 0 1 0 5
```

ADDITION (position + velocity)

If the jth element of velocity (vj) is equal to 0, the jth position value (xj,t) is inserted into the jth element of the new position (xj,t+1).
In the meantime, if vj is nonzero and does not appear in the new position, then xj,t+1=vj. Otherwise, xj,t+1 is equal to 0. For example,

```
X1,t = 3 2 5 1 4
V1,t = 0 2 0 3 4
X1,t+1 = 3 2 5 0 4
```

ADDITION (velocity + velocity)

For new velocity, V=V1 + V2, the jth element of V can be derived as follows,
Vj = v1,j IF v1,j!=0 && v2,j!=0
Vj = v1,j IF v1,j!=0 && v2,j==0
Vj = v2,j OTHERWISE
For example,

```
V1,t = 3 2 0 1 0
V2,t = 0 1 5 3 0
Vt+1 = 3 2 5 1 0
```

MULTIPLICATION (coefficient * velocity)

This operation can be represented as V2= c * V1, where coefficient c lies in[0, 1] is used to control the effect of V1 that inherits in V2.
For this purpose, a random number, r that lies in [0, 1] is generated. IF r< c, v2=v1, ELSE v2= 0.
For example,

```
V1 = 2 4 1 3 5, C = 0.4
R = .1 .7 .3 .9 .3
V2 = 2 0 1 0 5
```

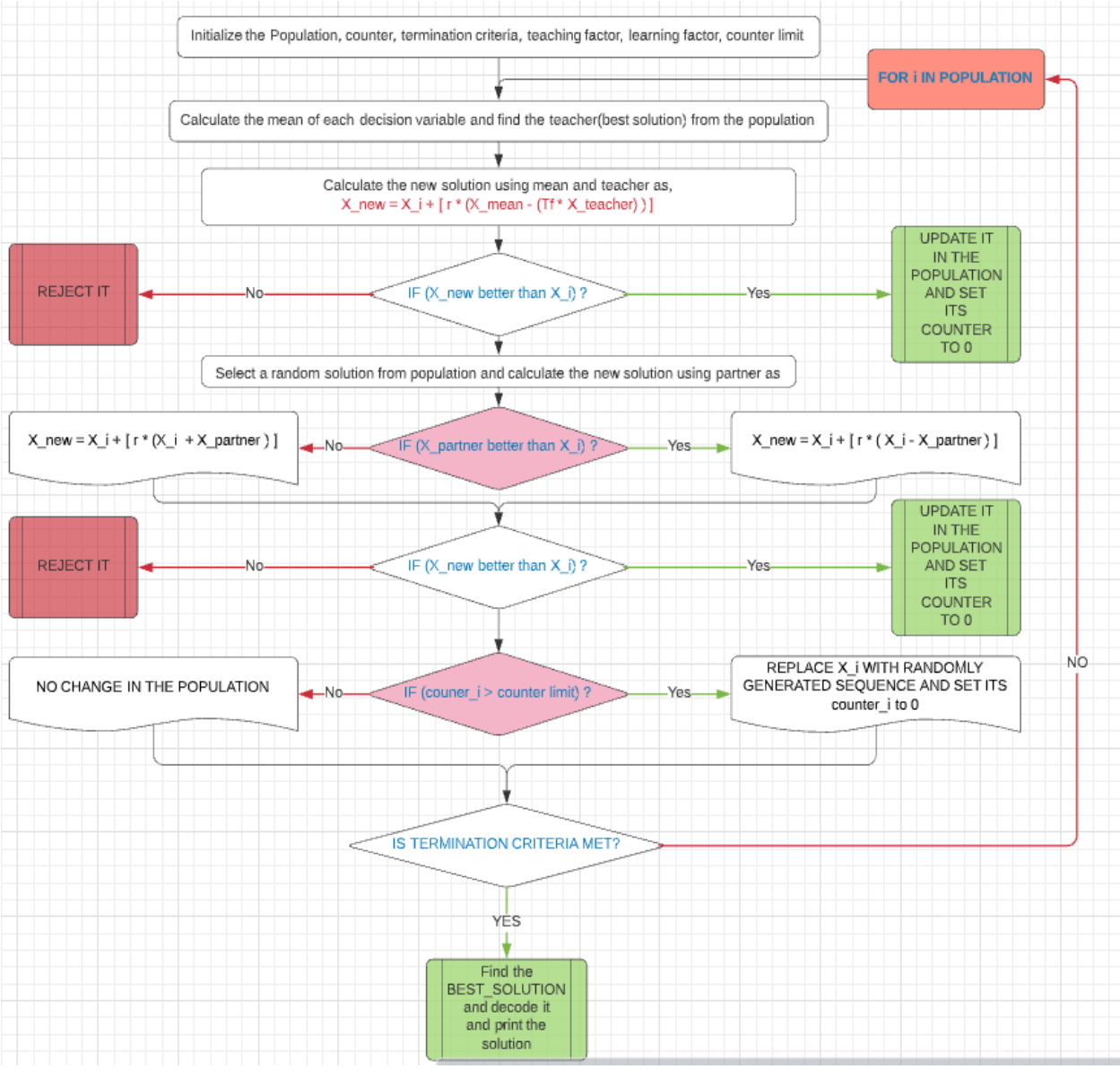
REPAIR FUNCTION

This function is used to repair a sequence which does not follow the precedence relation. This does multiple swaps and replacements so that the given sequence is in bounds with the precedence relation.
For example,

```
X = 0 3 1 4 5
new_X = 1 2 3 4 5
```

7.2 WORKING OF HYBRID TLBO:

PROPOSED FLOWCHART:



NOTATIONS USED:

X_i = a single i'th solution in the population.

r = Coefficient that indicates the impact of the mean of the population and the teacher

Tf = Teaching coefficient that indicates the impact of teacher

X_new = newly generated solution for the ith solution in the population

X_mean = mean of the population

X_best = teacher (best solution in the population)

X_partner = mean of the population

CALCULATING THE MEAN OF THE POPULATION

- As mentioned in the previous slide, in teacher phase calculation of the mean of the population is used in the equation, **X_new = X_i + r * [X_mean - (Tf * X_best)].**
- As the traditional way of calculating the mean can't be used here (discrete optimization problem), a different methodology is used here.
- The calculation is done as,
  - First the average of the task numbers in that j'th location in the population is calculated.
  - If the calculated average is not already present in the mean sequence, then it is pushed into the mean sequence.
  - If the calculated mean value is already present in the mean sequence, then the closest value to the mean in that j'th location of the population is inserted if it is not present in the mean sequence.
  - Otherwise, a random number is generated within the range of 1 - n (number of tasks) and is inserted in the mean sequence if it is not already present in the mean sequence.

SUBTRACTION OPERATOR

- As mentioned previously, in teacher phase and learner phase, subtraction of the solutions is done as,

  X_new = X_i + r * [X_mean - (Tf * X_best)].
- As the traditional way of subtraction is not possible here as the variables are discrete, a different methodology is used as,

  - For each task number in the i'th position of the sequences, if both the sequence have the same task number, then 0 is inserted in the i'th position of the result.
  - If they are not the same, a random number is generated between 0 and 1 and is compared with the coefficient(Tf / Tp).
  - If the random number generated is less than the coefficient then, the task number in the i'th position from the second sequence is inserted into the i'th position of the result.
  - Otherwise, 0 is inserted in the i'th position of the result.

ADDITION OPERATOR

- As mentioned previously, in the teacher phase and learner phase, addition of the solutions is done as,

  X_new = X_i + r * [X_mean - (Tf * X_best)].
- As the traditional way of addition is not possible here as the variables are discrete, a different methodology is used as,

  - For each task number in the i'th position of the sequences, if both the sequence have the same task number, then the task number from the i'th position of the first sequence is inserted in the i'th position of the result.

- ➔ If they are not the same, a random number is generated between 0 and 1 and is compared with the coefficient(r).
- ➔ If the random number generated is less than the coefficient, then the task number in i'th from the second sequence is inserted into the i'th position of the result.
- ➔ Otherwise, 0 is inserted in the i'th position of the result.

7.3 GENERAL FLOW OF MIGRATING BIRDS OPTIMISATION (MBO) ALGORITHM:

Input: Algorithm parameters and instance data
% Initialization
Initialize the *n* solutions randomly and place the population in the V-shape;
Do
For *i*=1 ***to*** *m* **do**
% Leader improvement
Obtain *k* neighbor solutions of the leader individual;
Update the leader individual with the best neighbor solution when the best neighbor solution achieves better fitness;
% Block improvement
For each individual in the left and right sides
Obtain (*k*-*x*) neighbor solutions of this individual;
Update this individual with the best one of the its (*k*-*x*) neighbor solutions and *x* unused best neighbor solutions in the front when the best neighbor solution achieves better fitness; **%Benefit mechanism**
Endfor
Endfor
% Leader replacement
Remove the leading individual to the end of either side;
Forward the following individual to the leading position as the new leader;
Until (termination criterion is satisfied)
Output: Best objective value

7.3.1 PHASES IN MBO ALGORITHM:

- **LEADER IMPROVEMENT:**

> **% Leader improvement**
> Obtain *k* neighbor solutions of the leader individual;
> Update the leader individual with the best neighbor solution when the best neighbor solution achieves better fitness;

First the leader is found using NDS(non dominated sorting) & CD (crowding distance) similar to PSO'S g_best.
To obtain neighbour solutions, two methods are proposed. They are,

1. By changing the task sequence (k/3)
2. By changing the process alternative sequence (k/3)
3. By changing both (k/3)

The value of k is fixed as, k = 10%-20% of the population size.

- **BLOCK IMPROVEMENT:**

> **% Block improvement**
> **For** each individual in the left and right sides
> Obtain (*k*-*x*) neighbor solutions of this individual;
> Update this individual with the best one of the its (*k*-*x*) neighbor solutions and *x* unused best neighbor solutions in the front when the best neighbor solution achieves better fitness; **%Benefit mechanism**

Each individual in the left and right just means that, solutions other than g_best.
To obtain neighbour solutions, the same methods as mentioned before are used. They are,

1. By changing the task sequence ((k-x)/3)
2. By changing the process alternative sequence ((k-x)/3)
3. By changing both ((k-x)/3)

The value of x here is fixed as any number less than k. Can be generated randomly or fixed.

- **CHANGING THE TASK SEQUENCE PROCESS ALTERNATIVES**

To change and find neighbor solutions of a particular solution,

➢ First a random number **'r'** is generated between (1 - n), where n = no. of tasks.
➢ Then the tasks in position (1 - r) are kept unchanged and the tasks from position (r+1 - n) are shuffled randomly such that precedence relation is followed. For example, (taking 11 tasks)

BEFORE CHANGING - 1 3 4 7 8 2 6 9 5 11 10
RANDOM NUMBER GENERATED (r) - 4
AFTER CHANGING - 1 3 4 7 6 9 8 2 5 10 11

➢ To change and find neighbor solutions of a particular solution, SWAPPING and SINGLE POINT MUTATION (randomly picking a position and changing its value) is performed such that the constraints are satisfying. For example for a 4 station problem,

BEFORE CHANGING - 1 3 1 2
AFTER PERFORMING SWAP - 1 3 2 1 (last two positions are swapped)
AFTER SINGLE POINT MUTATIONS - 2 3 2 1 (1st position is mutated)

- **LEADER IMPROVEMENT:**

% Leader replacement

Remove the leading individual to the end of either side;

Forward the following individual to the leading position as the new leader;

➢ This leader replacement phase removes the leader and replaces it with a better candidate from either side.
➢ This helps the flock of birds to head to better direction/solution space and arrive at a better pareto front.

7.4 GENERAL WORKING AND FLOW OF ARCHIMEDES OPTIMISATION ALGORITHM (AOA):

➢ The AOA or Archimedes optimisation algorithm works on the basis of Archimedes principle, where buoyant force is exerted on objects partially immersed in water, to maintain their equilibrium.
➢ Assuming multiple objects immersed in the same fluid, where the objects are analogous to each population member and the fluid media is the search space, each object(member) tries to reach equilibrium (optimised state), which is if the respective buoyant force Fb is equal to the their individual weights Wr.

MATHEMATICAL FORMULAE USED:

$$F_b = W_o,$$
$$p_b v_b a_b = p_o v_o a_o$$
$$a_o = \frac{p_b v_b a_b}{p_o v_o}$$

$$W_b - W_r = W_o,$$
$$p_b v_b a_b - p_r v_r a_r = p_o v_o a_o$$

p → Density;

v → Volume;

a → acceleration due to gravity;

Subscripts: b for fluid and o for each immersed object.

} The equilibrium state equation used when there is another force influenced on the object like collision with another neighbouring object (r)

Initialize a random population, velocity, density, volume, acceleration

For t=1 to T (termination criteria)

For i=1 to P (population size)

1. **UPDATE density ([1 x P] array) and volume ([1 x P] array)**
2. **UPDATE TF (transfer operator) & d (density decreasing factor)**
3. **UPDATE acceleration ([1 x P] array)**
4. **NORMALISE acceleration**
5. **UPDATE POPULATION using the updated parameters**

END

END

7.4.1 PHASES IN AOA ALGORITHM:

- **UPDATION OF DENSITY AND VOLUME:**

➢ Initial density and volume for each object of the population is generated randomly as a variable between **[0-1].** Then for next iterations, the equations:

$$den(i)_{t+1} = den(i)_t + r * (den_{best} - den(i)_t)$$
$$vol(i)_{t+1} = vol(i)_t + r * (vol_{best} - vol(i)_t)$$

where,
$den(i)_t$ → density of i'th object in t'th iteration
den_{best} → density of best object in that population
$vol(i)_t$ → density of i'th object in t'th iteration
Vol_{best} → density of best object in that population
r → random number between (0-1)

- **UPDATION OF Tf and d:**

➢ The TF and d are the fine tuning parameters, which are calculated as the iteration progresses, rather than tuned by the user.
This makes this algorithm to be more randomized and exploit the search space, without having any user-defined bias.

TF_t = exp ((t - t_max) / t_max)
d_t = exp ((t_max - t) / t_max)) - (t / t_max)

where,
TF_t → transfer operator for t'th iteration
t_max → maximum number of iterations
d_t → density decreasing factor for t'th iteration

- **UPDATION OF ACCELERATION:**

➢ Initial acceleration for each object of the population is generated randomly as a variable between **[0-1].** Then for next iterations, the equation:

acc_i = ((den_r + vol_r * acc_r) / (den_i * vol_i) **IF TF <= 0.5**
(exploration phase - when collision between object occurs)

acc_i = ((den_best + vol_best * acc_best) / (den_i * vol_i) **TF > 0.5**
(exploitation phase - when there is no collision between objects)

where,
acc_i → acceleration of i'th object
den_best → density of best object in that population
den_r → density of r'th object ('r' is generated randomly)
vol_best → volume of best object in that population
vol_r → volume of r'th object ('r' is generated randomly)
acc_best → acceleration of best object in that population
acc_r → acceleration of r'th object ('r' is generated randomly)

- **NORMALISATION OF ACCELERATION:**

➢ The updated acceleration of each object is then normalised as:

acc_i(norm) = (u * (acc_i - acc_min) / (acc_max - acc_min))) + l

Where,
acc_i(norm) → normalised acceleration of i'th object
acc_i → acceleration of i'th object in that population
Acc_max → maximum acceleration in the population
acc_min → minimum acceleration in the population
u = 0.9 (range of normalisation)
l = 0.1 (range of normalisation)

- ➢ The acc_i(norm) determines the percentage of steps that each object will change.
- ➢ If the object 'i' is far away from global optimum, acceleration value will be high—meaning that the object will be in the exploration phase, otherwise, in the exploitation phase.
- ➢ This helps search agents move towards the global best solution and at the same time they move away from local solutions.

- **UPDATION OF POPULATION:**

- ➢ IF **TF<=0.5** - exploration phase equation is called (collision between objects)
  $x(i)_{t+1} = x(i)_t + (d * acc(i)_{t+1} * (x(r)_t - x(i)_t))$
- ➢ IF **TF>0.5** - exploitation phase equation called (no collision)
  $x(i)_{t+1} = x(i)_t + (F * d * acc(i)_{t+1} * (x\_best - x(i)_t))$

  where,
  $acc(i)_t$ → acceleration of i'th object in t'th iteration
  $x(i)_t$ → decision variable of i'th object in t'th iteration
  x_best → best value of decision variable in the population
  $x(r)_t$ → decision variable of r'th object in t'th iteration where 'r' is generated randomly
  d → density decreasing factor
  F → Randomly generated, either -1 or +1

7.4.2 WHY AOA ALGORITHM FOR OUR H-R ALB PROBLEM:

- ➢ **A recent algorithm (2020-2021) that has never been explored much. This algorithm has never been used in assembly line balancing problems which gives more value and novelty to our paper/project.**
- ➢ **Has dedicated exploration and exploitation phases that creates a balance between exploration and exploitation.**
- ➢ **Has no fine tuning parameters.**
- ➢ **Any type of user with no knowledge of metaheuristic algorithms can use this algorithm and get results.**
- ➢ **Faster compared to other algorithms as computations of simple math equations are faster.**
- ➢ **Not much complexity involved in updating the population.**
- ➢ **Larger search space is explored.**
- ➢ **Easy to implement and hybridize with other algorithms.**

7.5 HYBRIDISATION PROCEDURE FOR OUR ALGORITHMS OF TLBO, MBO, PSO AND AOA:

- ❖ Variants for algorithms are created to make the algorithm give better results.
- ❖ By hybridizing the algorithm with another combines the advantages of both the algorithm and gives a better solution than a single algorithm.
- ❖ The algorithm chosen to hybridize with the developed 4 algorithms is ABC (artificial bee colony algorithm).
- ❖ The scout phase from this ABC algorithm is added with those 4 algorithms(PSO, TLBO, MBO, AOA).
- ❖ The main reason for adding the scout phase is that it increases the exploration and exploitation space and gives better pareto fronts by replacing the retained solutions and replacing it with a random one.
- ❖ Easy to implement and hybridize with other algorithms.
- ❖ All these algorithms are compared using those performance indicators. The best performing algorithm after testing those with different data sets is presented in the next section.

```
/* Initialization */
Site initial food sources using (1)
Cycle = 1
For each food source(i)
   unsuccessfulTrials(i) = 0
End
Repeat
  /* Employed Bees Operator*/
  For each employed bee(i)
       Send employed bee to corresponding food source(i)
       Site a new food source using (2)
       Apply greedy selection
       If new food source selected
          unsuccessfulTrials(i) = 0
        Else
          unsuccessfulTrials(i) = unsuccessfulTrials(i) + 1
  End
  /* Onlooker Bees Operator */
  Calculate probability P_i for each food source(i) using (3)
  For each food source(i)
       Send onlooker bees to food source(i) in accordance
       with associated P_i
       For each onlooker bee(j)
          Site a new food source using (2)
          Apply greedy selection
          If a new food source selected
             unsuccessfulTrials(i) = 0
          Else
             unsuccessfulTrials(i) = unsuccessfulTrials(i) + 1
         End
  End
  /* Scout Bee Operator */
  For each food source(i)
       If unsuccessfulTrials(i) > limit
          Abandon the food source(i)
          Site a new food source using (1)
  End
  Memorize the best solution achieved so far
  Cycle = Cycle + 1
Until termination criteria are met
```

7.6 PERFORMANCE INDICATORS (PI's) USED FOR COMPARISON:

Other than illustrative/graphical plots of performance in terms of optimal parameters to yield optimal solutions, a set of five performance indicators as shown below, represent a quantitative way of comparing performance results with respect to every crucial mechanisms involved in multi-objective optimisation technique used (NSGA - II) as well as the different algorithms.

PI 1 - NUMBER OF NON DOMINATED SOLUTIONS FOUND

- ➔ The number of non-dominated solutions generated by each algorithm in Pareto solution.
- ➔ Higher PI (1) shows better algorithm performance.

PI 2 - ERROR RATIO

- ➔ The number of solutions which are not members of the Pareto optimal set divided by the number of solutions generated by the algorithm.
- ➔ Smaller PI (2) shows better algorithm performance.

PI 3 - CONVERGENCE METRIC

- ➔ This metric measures the extent of convergence to a known set of Pareto optimal solutions, as follows:

$$\Upsilon = \frac{\sum_{i=1}^{N} d_i}{N},$$

where, **N** is the number of nondominated solutions obtained with an algorithm and, **di** is the Euclidean distance between each of the nondominated solutions and the nearest member of the true Pareto optimal front.

PI 4 - SPACING

- ➔ This indicator measures the relative distance between each solution.
- ➔ where, **di** is the distance between solution **i** and the nearest solution, while **d** is the average of all **di**.
- ➔ The smaller spacing index shows a better solution and has better space between each solution.

$$Spacing = \sqrt{\frac{1}{N}\sum_{i=1}^{s_q}(d_i - \bar{d})^2}$$

PI 5 - MAXIMUM SPREAD

- ➔ The spread of solutions found by each algorithm.
- ➔ Larger maximum spread is better.

$$Spread_{max} = \sqrt{\sum_{i=1}^{M} (\min f_i - \max f_i)^2}$$

CHAPTER 8

RESULTS & DISCUSSIONS

8.1 FINE TUNING OF ITERATIONS AND POPULATION SIZE:

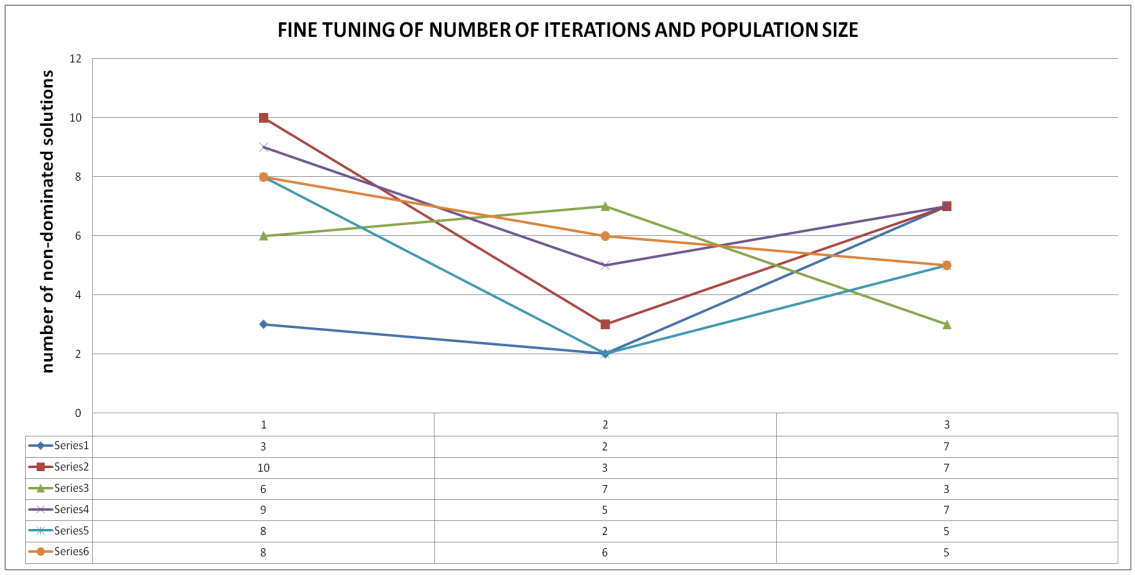
- ➢ The number of iterations and population size is actually given by the user, but for different sized data, the population size and iterations must be varied so that the results obtained are optimal.
- ➢ To do this, all the algorithms are tested with different sized datasets namely,
  - ➔ **Small -** 0 < Number of tasks <= 50
  - ➔ **Medium -** 50 < Number of tasks <= 100
  - ➔ **Large -** Number of tasks > 100
- ➢ The obtained results are compared using the performance indicators and the number of iterations and population size are fixed accordingly.
- ➢ The sizes of iterations and population are considered the same to get better results, from experimental trials run for a number of combinatorial sizes. The various types tested with are,
  - ★ **Series 1 -** T = 5 , P = 5
  - ★ **Series 2 -** T = 10, P = 10
  - ★ **Series 3 -** T = 15, P = 15
  - ★ **Series 4 -** T = 20, P = 20
  - ★ **Series 5 -** T = 25, P = 25
  - ★ **Series 6 -** T = 30, P = 30

  Where,
  T = Number of iterations
  P = population size

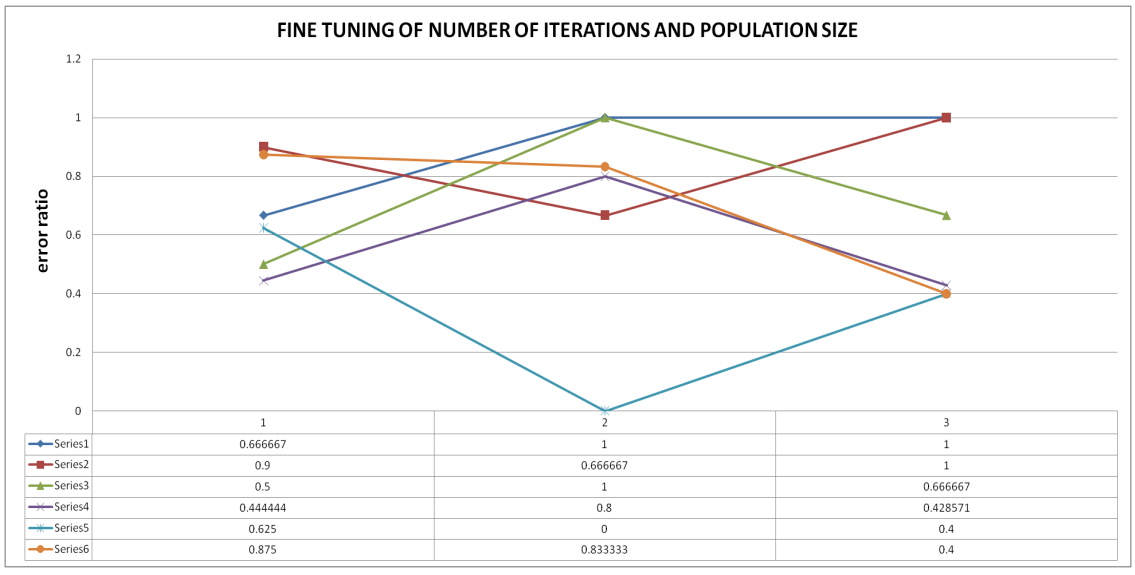
- ★ **PI 1 -** number of non dominated solutions
- ★ **PI 2 -** error ratio
- ★ **PI 3 -** convergence metric
- ★ **PI 4 -** spacing
- ★ **PI 5 -** maximum spread

8.1.1 RESULTS FOR PSO ALGORITHM:

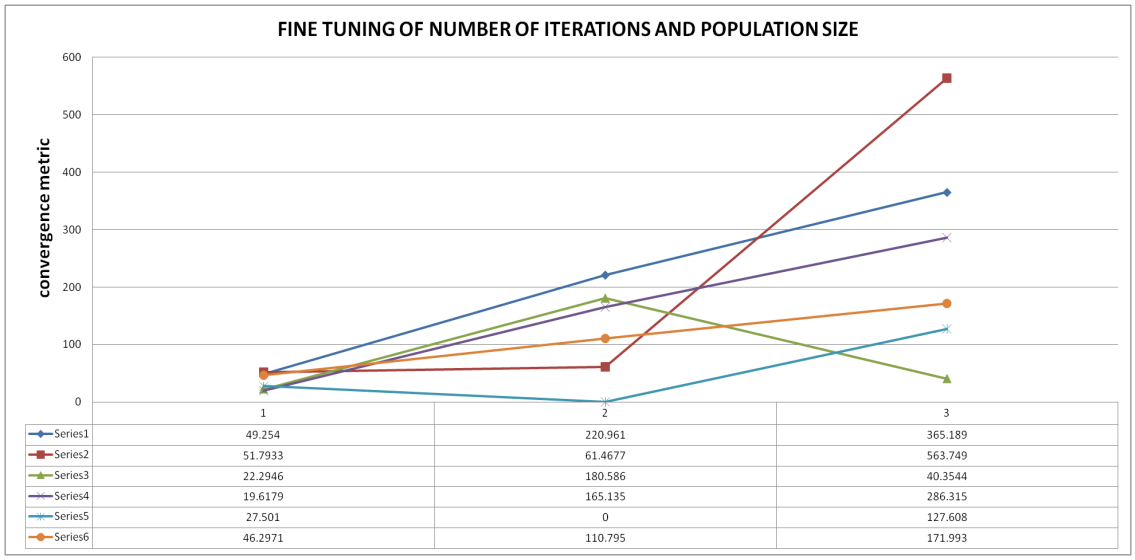
Following are the graphical plots for every performance indicator mentioned earlier VS the task dataset sizes- small, medium and large, represented by 1, 2 and 3 respectively for PSO algorithm alone.



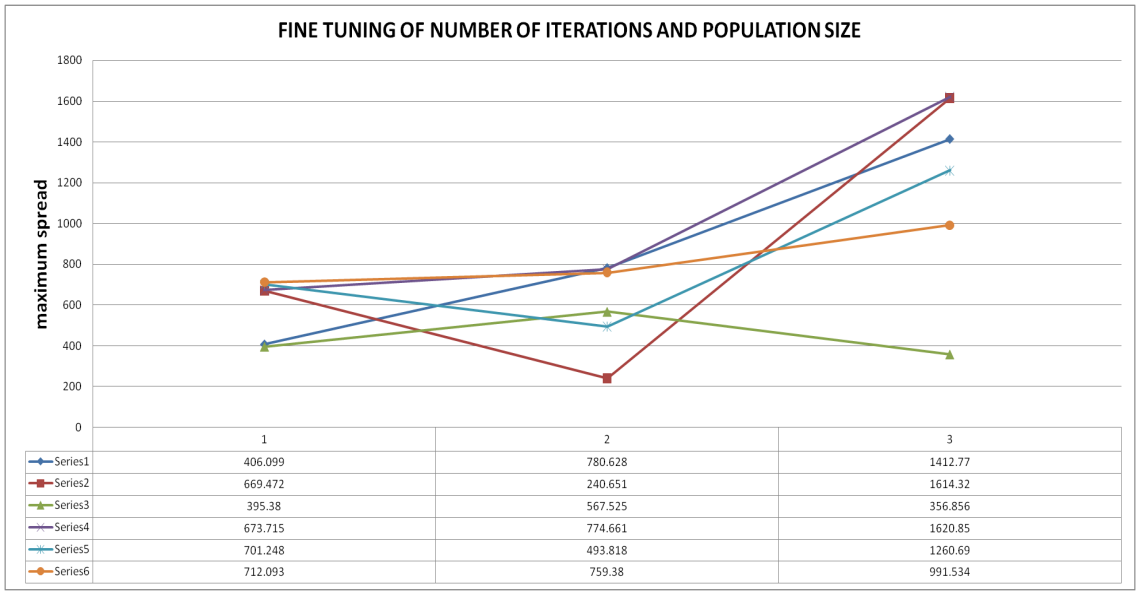
	1	2	3
Series1	3	2	7
Series2	10	3	7
Series3	6	7	3
Series4	9	5	7
Series5	8	2	5
Series6	8	6	5



	1	2	3
Series1	0.666667	1	1
Series2	0.9	0.666667	1
Series3	0.5	1	0.666667
Series4	0.444444	0.8	0.428571
Series5	0.625	0	0.4
Series6	0.875	0.833333	0.4



	1	2	3
Series1	49.254	220.961	365.189
Series2	51.7933	61.4677	563.749
Series3	22.2946	180.586	40.3544
Series4	19.6179	165.135	286.315
Series5	27.501	0	127.608
Series6	46.2971	110.795	171.993



	1	2	3
Series1	406.099	780.628	1412.77
Series2	669.472	240.651	1614.32
Series3	395.38	567.525	356.856
Series4	673.715	774.661	1620.85
Series5	701.248	493.818	1260.69
Series6	712.093	759.38	991.534

PROBLEM SIZE	PI 1	PI 2	PI 3	PI 4	PI 5	OVERALL
Small	series 2	series 4	series 4	series 1	series 6	**series 4**

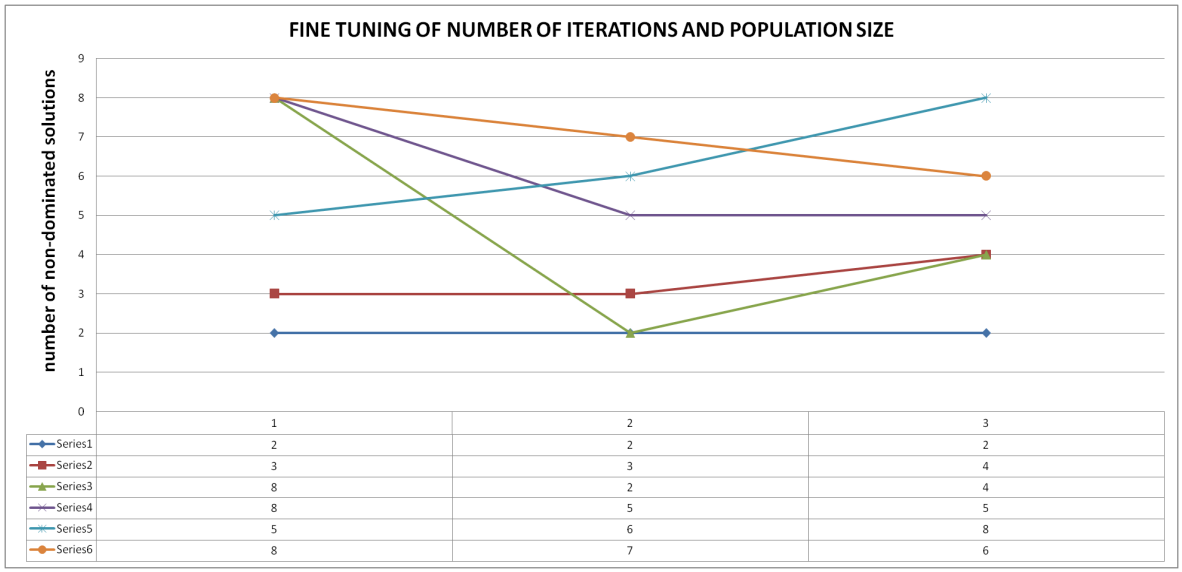
Medium	series 3	series 5	series 5	series 5	series 6	**series 5**
Large	series 4	series 6	series 3	series 6	series 4	**series 6**

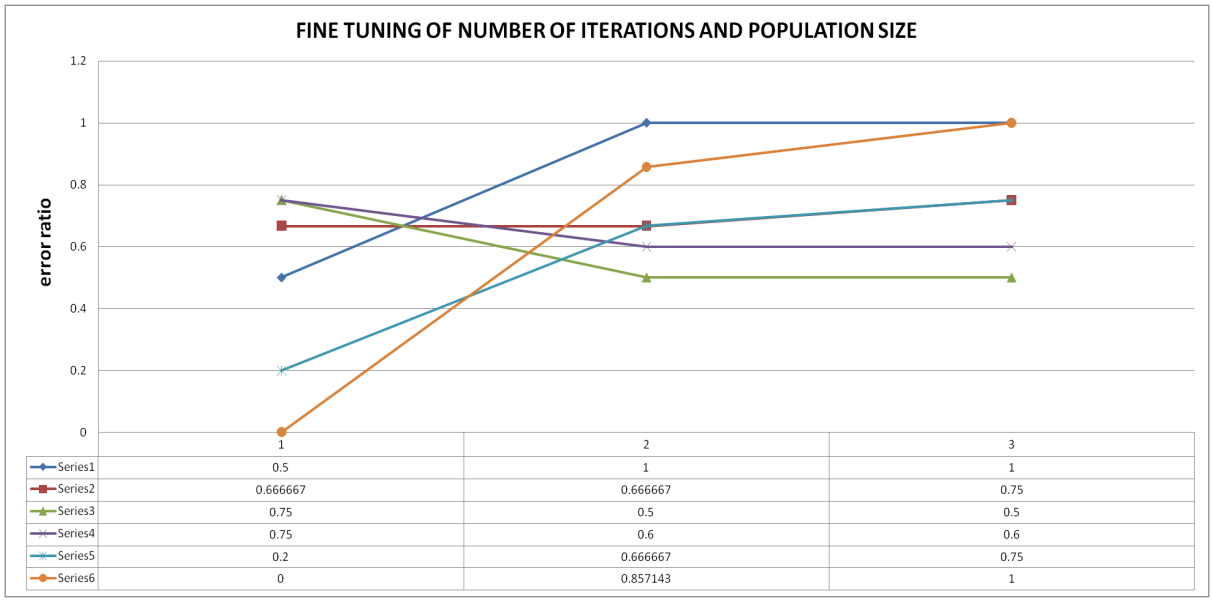
INFERENCE:

Based on the size of the task dataset, the results for the overall best performing "SERIES" range for PSO algorithm is determined based on the consolidated PI results.

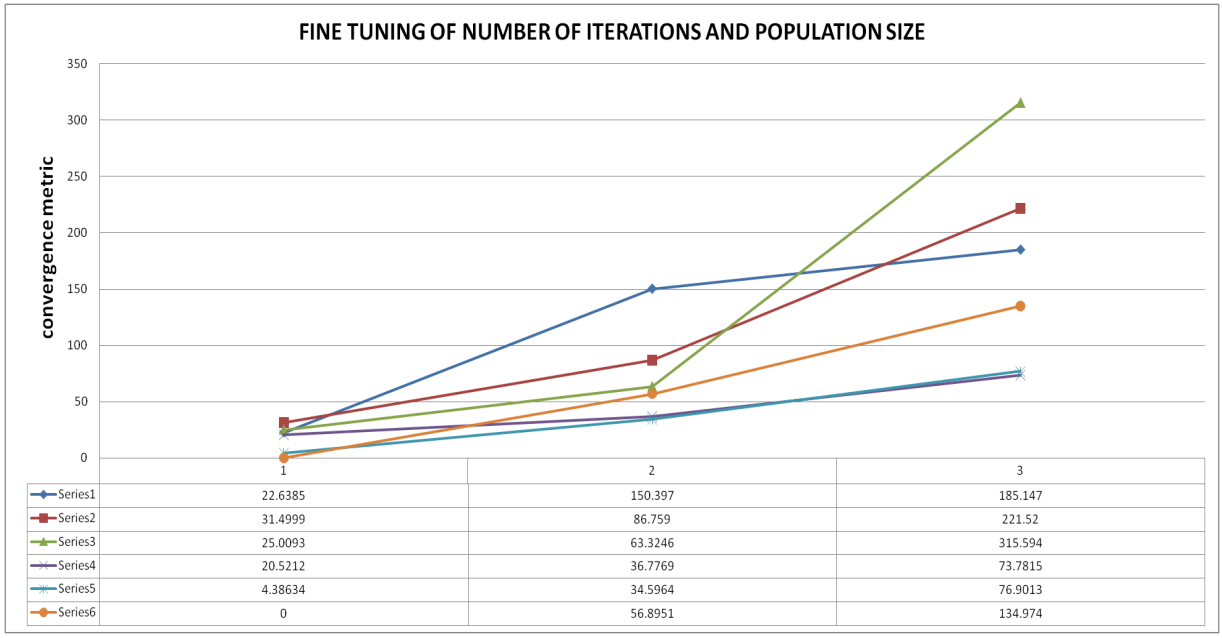
8.1.2 RESULTS FOR TLBO ALGORITHM:

Following are the graphical plots for every performance indicator mentioned earlier VS the task dataset sizes- small, medium and large, represented by 1, 2 and 3 respectively for TLBO algorithm alone.

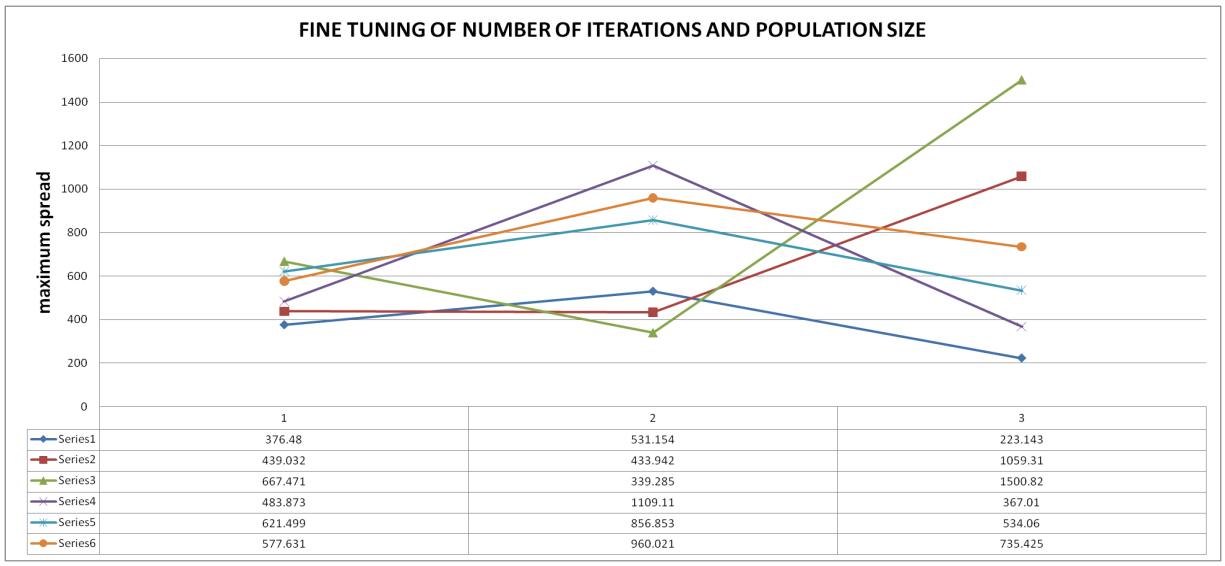




	1	2	3
Series1	0.5	1	1
Series2	0.666667	0.666667	0.75
Series3	0.75	0.5	0.5
Series4	0.75	0.6	0.6
Series5	0.2	0.666667	0.75
Series6	0	0.857143	1



	1	2	3
Series1	22.6385	150.397	185.147
Series2	31.4999	86.759	221.52
Series3	25.0093	63.3246	315.594
Series4	20.5212	36.7769	73.7815
Series5	4.38634	34.5964	76.9013
Series6	0	56.8951	134.974



	1	2	3
Series1	376.48	531.154	223.143
Series2	439.032	433.942	1059.31
Series3	667.471	339.285	1500.82
Series4	483.873	1109.11	367.01
Series5	621.499	856.853	534.06
Series6	577.631	960.021	735.425

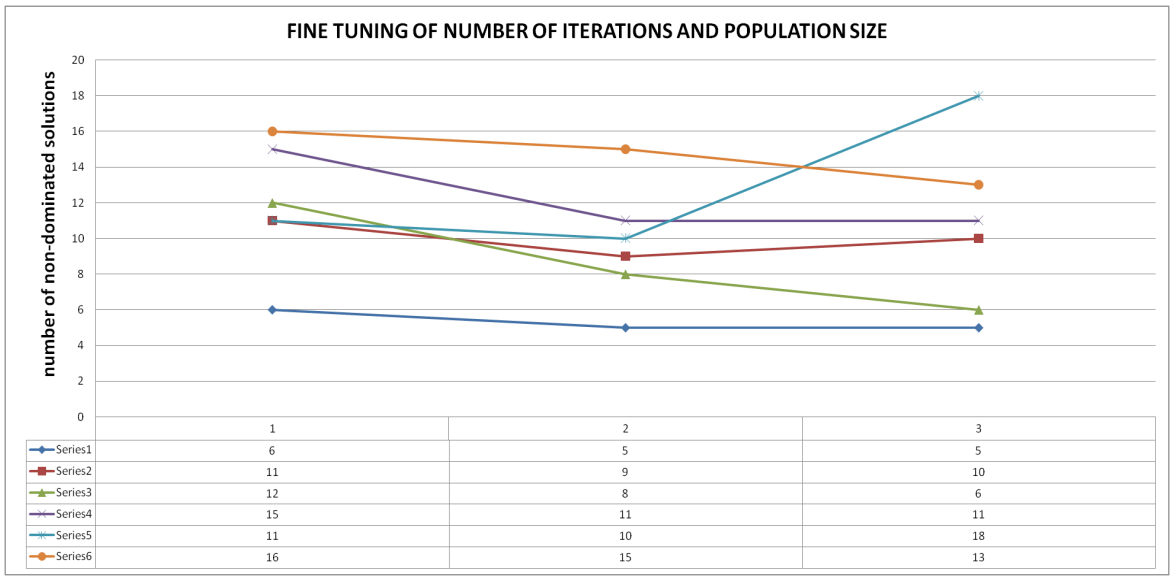
PROBLEM SIZE	PI 1	PI 2	PI 3	PI 4	PI 5	OVERALL
Small	series 3	series 5	series 5	series 4	series 3	**series 3**
Medium	series 6	series 3	series 5	series 6	series 5	**series 5**
Large	series 5	series 3	series 5	series 4	series 3	**series 5**

INFERENCE:

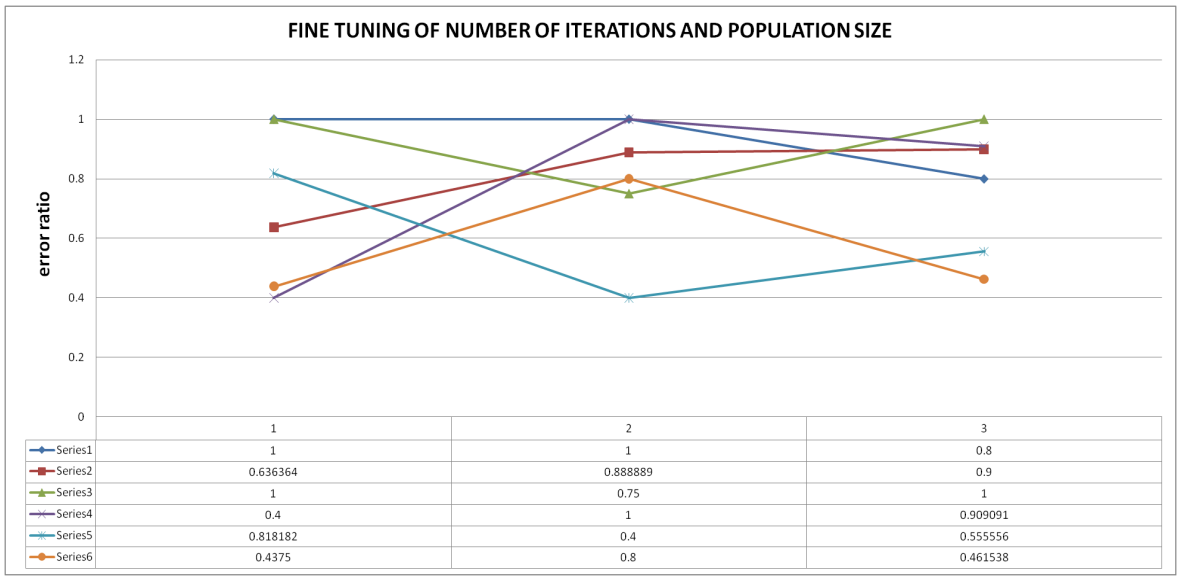
Based on the size of the task dataset, the results for the overall best performing “SERIES” range for TLBO algorithm is determined based on the consolidated PI results.

8.1.3 RESULTS FOR MBO ALGORITHM:

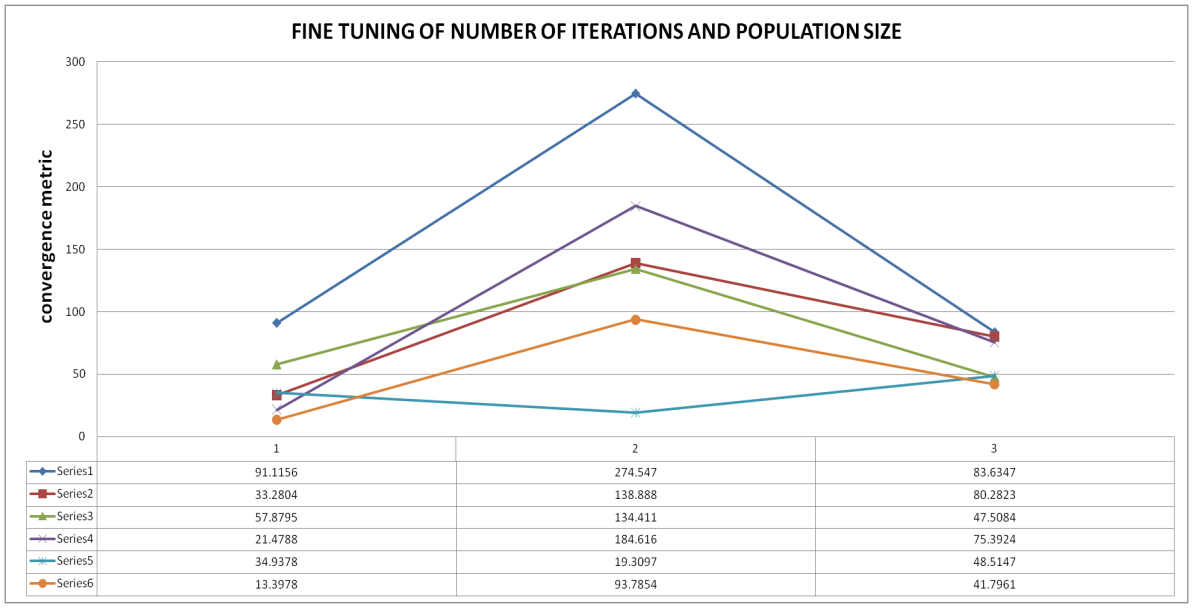
Following are the graphical plots for every performance indicator mentioned earlier VS the task dataset sizes- small, medium and large, represented by 1, 2 and 3 respectively for MBO algorithm alone.



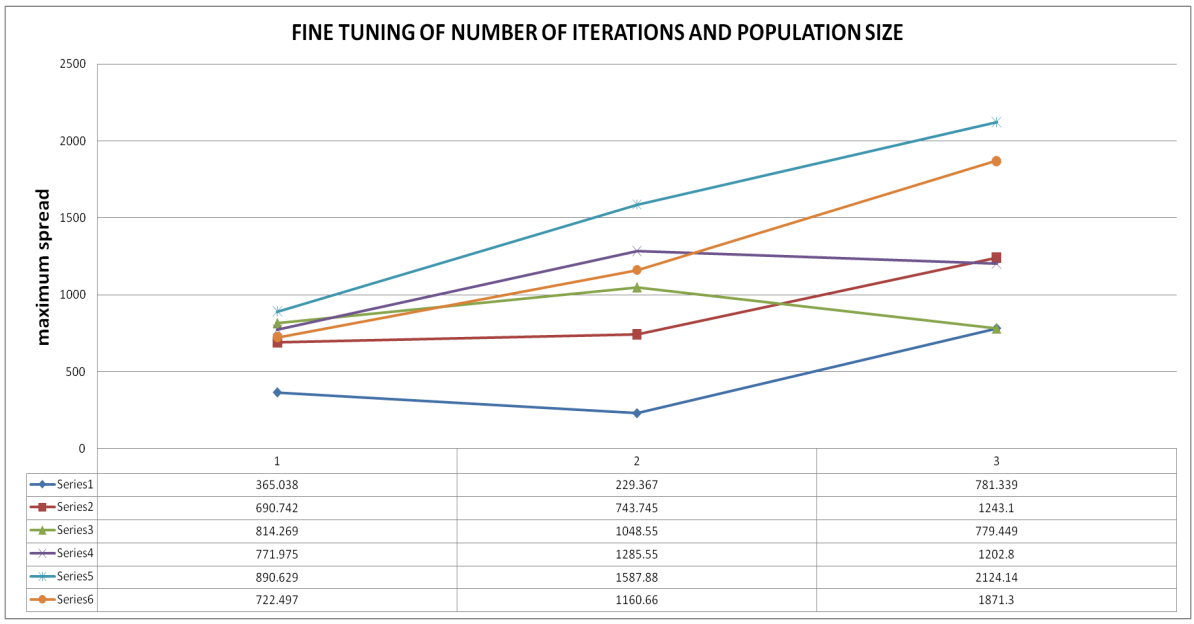
	1	2	3
Series1	6	5	5
Series2	11	9	10
Series3	12	8	6
Series4	15	11	11
Series5	11	10	18
Series6	16	15	13



	1	2	3
Series1	1	1	0.8
Series2	0.636364	0.888889	0.9
Series3	1	0.75	1
Series4	0.4	1	0.909091
Series5	0.818182	0.4	0.555556
Series6	0.4375	0.8	0.461538



	1	2	3
Series1	91.1156	274.547	83.6347
Series2	33.2804	138.888	80.2823
Series3	57.8795	134.411	47.5084
Series4	21.4788	184.616	75.3924
Series5	34.9378	19.3097	48.5147
Series6	13.3978	93.7854	41.7961



	1	2	3
Series1	365.038	229.367	781.339
Series2	690.742	743.745	1243.1
Series3	814.269	1048.55	779.449
Series4	771.975	1285.55	1202.8
Series5	890.629	1587.88	2124.14
Series6	722.497	1160.66	1871.3

PROBLEM SIZE	PI 1	PI 2	PI 3	PI 4	PI 5	OVERALL
Small	series 6	series 4	series 6	series 4	series 5	**series 4**

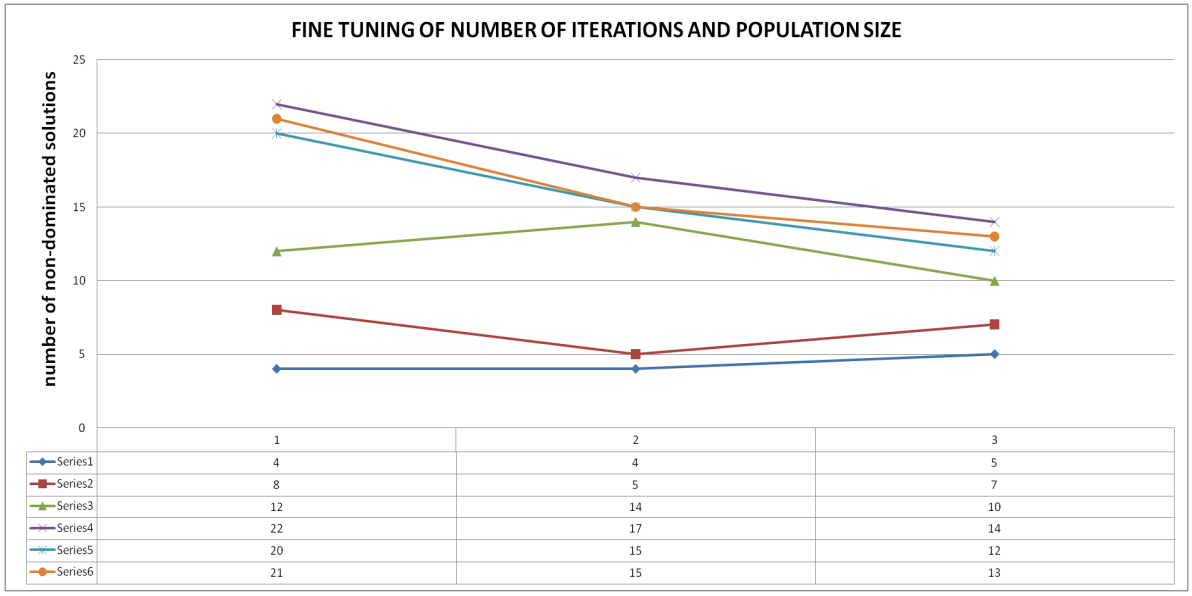
Medium	series 6	series 5	series 5	series 3	series 5	**series 5**
Large	series 5	series 6	series 6	series 3	series 5	**series 6**

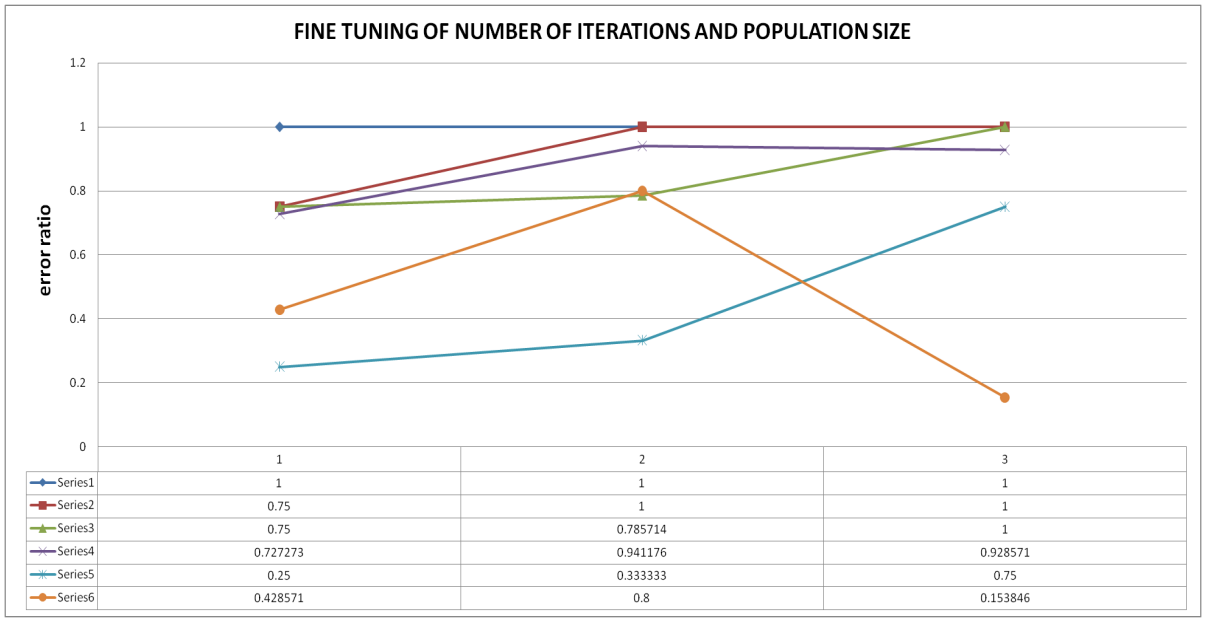
INFERENCE:

Based on the size of the task dataset, the results for the overall best performing "SERIES" range for MBO algorithm is determined based on the consolidated PI results.

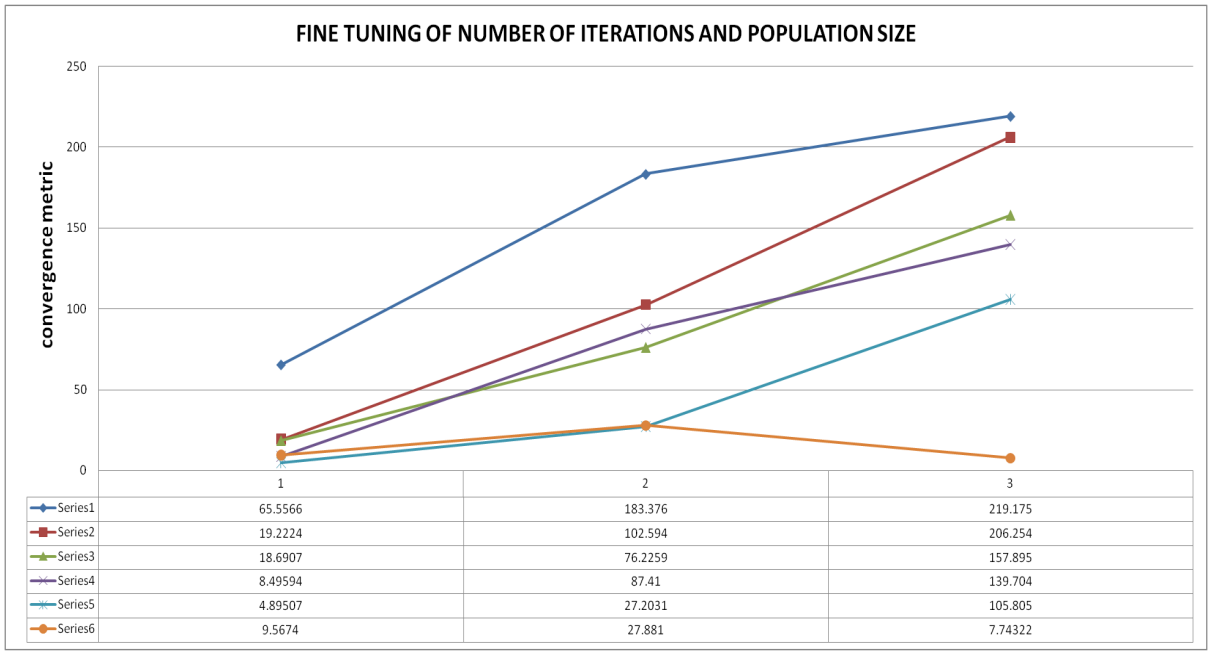
8.1.4 RESULTS FOR AOA ALGORITHM:

Following are the graphical plots for every performance indicator mentioned earlier VS the task dataset sizes- small, medium and large, represented by 1, 2 and 3 respectively for AOA algorithm alone.





	1	2	3
Series1	1	1	1
Series2	0.75	1	1
Series3	0.75	0.785714	1
Series4	0.727273	0.941176	0.928571
Series5	0.25	0.333333	0.75
Series6	0.428571	0.8	0.153846



	1	2	3
Series1	65.5566	183.376	219.175
Series2	19.2224	102.594	206.254
Series3	18.6907	76.2259	157.895
Series4	8.49594	87.41	139.704
Series5	4.89507	27.2031	105.805
Series6	9.5674	27.881	7.74322

PROBLEM SIZE	PI 1	PI 2	PI 3	PI 4	PI 5	OVERALL
Small	series 4	series 5	series 6	series 5	series 4	**series 5**

Medium	series 4	series 5	series 6	series 2	series 6	**series 6**
Large	series 4	series 6	series 6	series 6	series 5	**series 6**

INFERENCE:

Based on the size of the task dataset, the results for the overall best performing "SERIES" range for AOA algorithm is determined based on the consolidated PI results.

Based on the above results for individual algorithms, an overall result tabulation is shown.

ALGORITHM	PSO	TLBO	MBO	AOA
SMALL	T = 20, P = 20	T = 15, P = 15	T = 20, P = 20	T = 25, P = 25
MEDIUM	T = 25, P = 25	T = 25, P = 25	T = 25, P = 25	T = 30, P = 30
LARGE	T = 30, P = 30	T = 25, P = 25	T = 25, P = 25	T = 30, P = 30

These results can be inferred by researchers working on the improvement of the same algorithms as above, basing their iterations and population sizes according to the size of the problem considered by them.

8.2 FINE TUNING OF PARAMETERS INVOLVED IN VARIOUS ALGORITHMS:

- For all the algorithms developed in our project, their respective parameters are fine tuned using certain efficient criteria as described below.
- Thus for each algorithm, based on the size category(small, medium, large) of task data sets are graphs and tabulations of the "fine tuning criteria". The best criteria is then tabulated based on the performance indicator values.
- The given criteria for each algorithm are given in the respective algorithm sections.
- For each algorithm, the fine tuning criterion for the various parameters are such that, the range of each parameter is split and given constraints, according to the significance of the

individual parameter in their algorithmic equations.

- ❖ These fine tuning criteria that are represented in ranges, are then determined for their ideal values using performance indicators analytically.

8.2.1 RESULTS FOR PSO ALGORITHM:

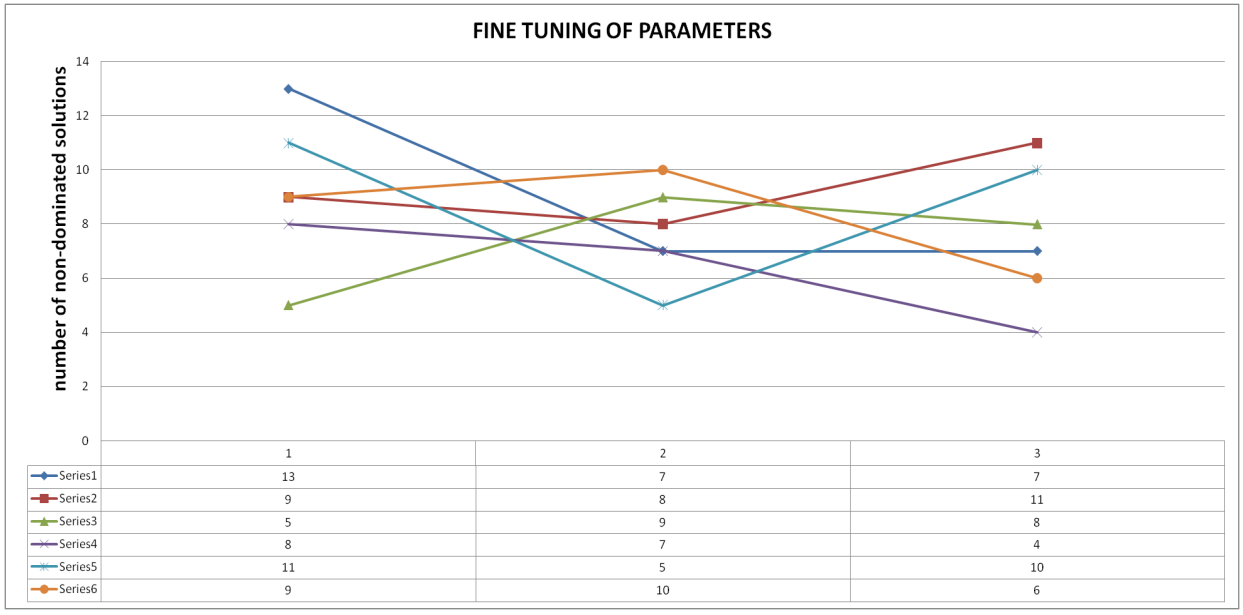
$$X_i^{t+1} = X_i^t + V_i^{t+1}$$

$$V_i^{t+1} = c_1 V_i^t + c_2(Pbest_i^t - X_i^t) + c_3(Gbest_t - X_i^t)$$

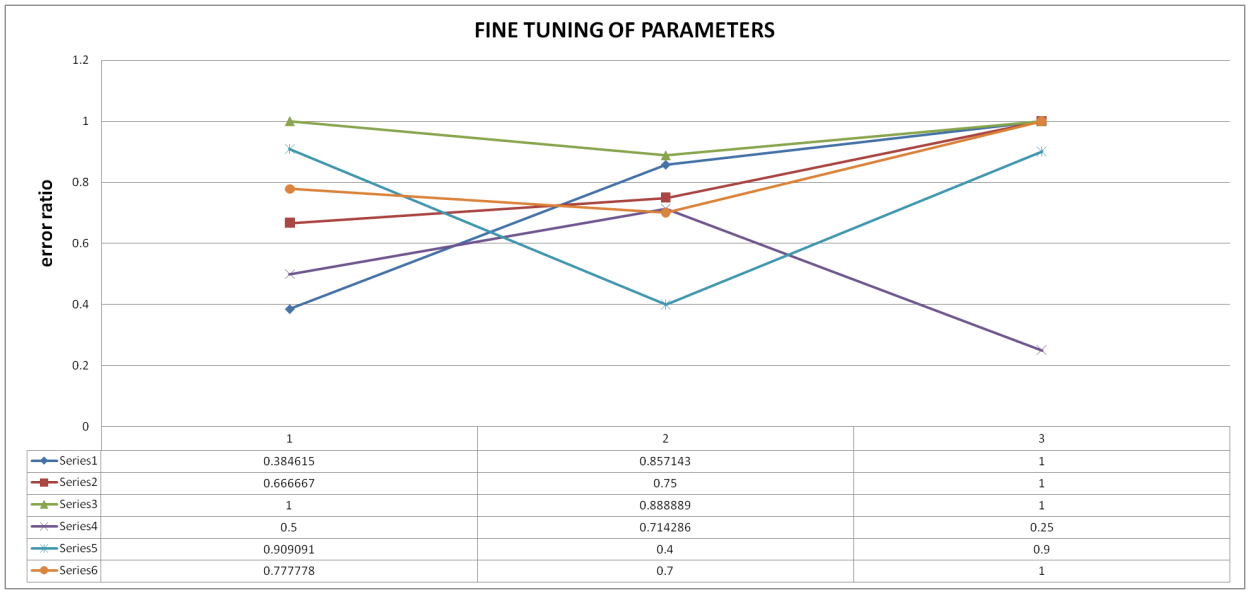
Where,
c1 = inertia weight
c2 = acceleration coefficient of P_best
c3 = acceleration coefficient of G_best

- ★ **Series 1 -** c1 ∈ (0, 0.5), c2+c3 <= 1.5
- ★ **Series 2 -** c1 ∈ (0.5, 1), c2+c3 <= 1.5
- ★ **Series 3 -** c2 ∈ (0, 0.5), c1+c3 <= 1.5
- ★ **Series 4 -** c2 ∈ (0.5, 1), c1+c3 <= 1.5
- ★ **Series 5 -** c3 ∈ (0, 0.5), c1+c2 <= 1.5
- ★ **Series 6 -** c3 ∈ (0.5, 1), c1+c2 <= 1.5

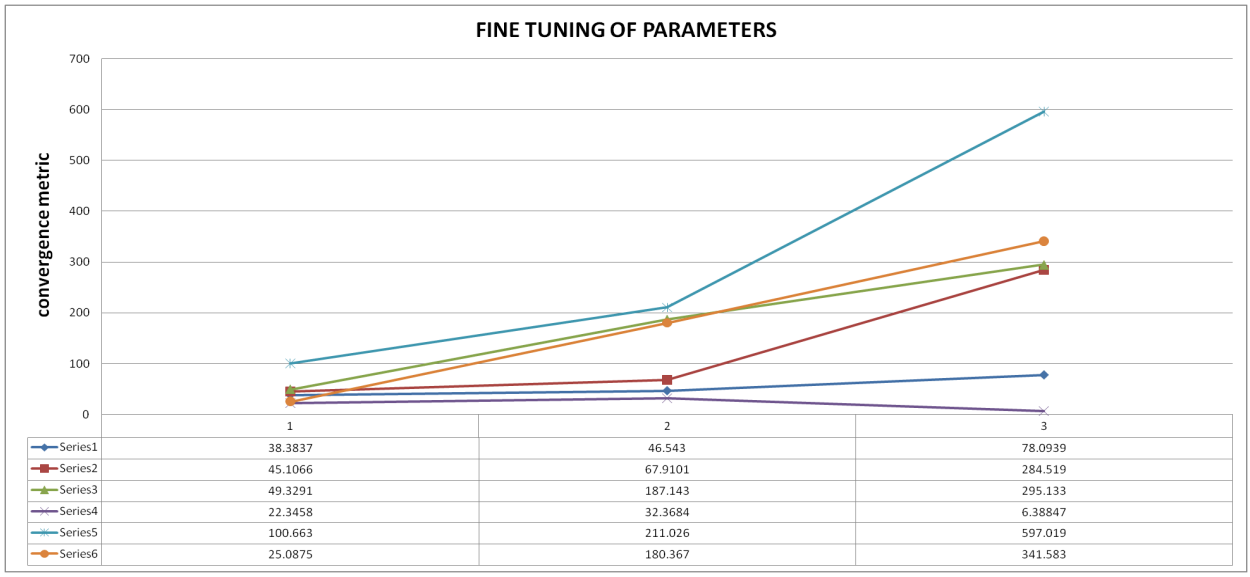
Following are the graphical plots for every performance indicator mentioned earlier VS the task dataset sizes- small, medium and large, represented by 1, 2 and 3 respectively for PSO algorithm alone.



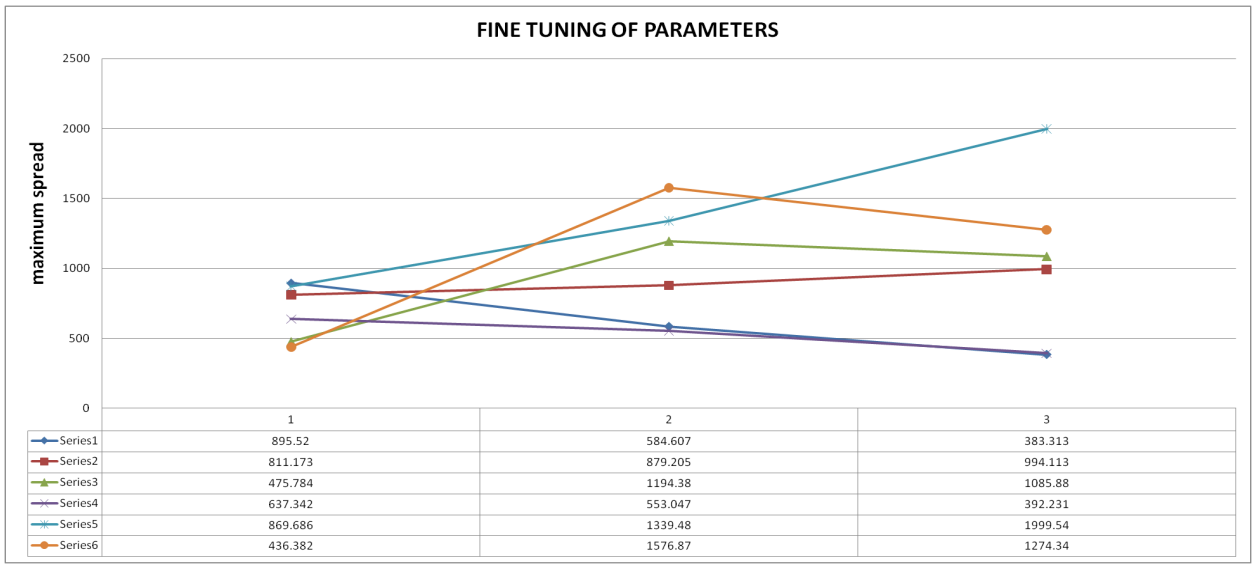
	1	2	3
Series1	13	7	7
Series2	9	8	11
Series3	5	9	8
Series4	8	7	4
Series5	11	5	10
Series6	9	10	6



	1	2	3
Series1	0.384615	0.857143	1
Series2	0.666667	0.75	1
Series3	1	0.888889	1
Series4	0.5	0.714286	0.25
Series5	0.909091	0.4	0.9
Series6	0.777778	0.7	1



	1	2	3
Series1	38.3837	46.543	78.0939
Series2	45.1066	67.9101	284.519
Series3	49.3291	187.143	295.133
Series4	22.3458	32.3684	6.38847
Series5	100.663	211.026	597.019
Series6	25.0875	180.367	341.583



	1	2	3
Series1	895.52	584.607	383.313
Series2	811.173	879.205	994.113
Series3	475.784	1194.38	1085.88
Series4	637.342	553.047	392.231
Series5	869.686	1339.48	1999.54
Series6	436.382	1576.87	1274.34

PROBLEM SIZE	PI 1	PI 2	PI 3	PI 4	PI 5	OVERALL
Small	series 1	series 1	series 4	series 5	series 1	**series 1**
Medium	series 6	series 5	series 4	series 4	series 6	**series 6**
Large	series 2	series 4	series 4	series 1	series 4	**series 4**

INFERENCE:

Based on the size of the task dataset, the results for the overall best performing “SERIES” range for PSO algorithm is determined based on the consolidated PI results.

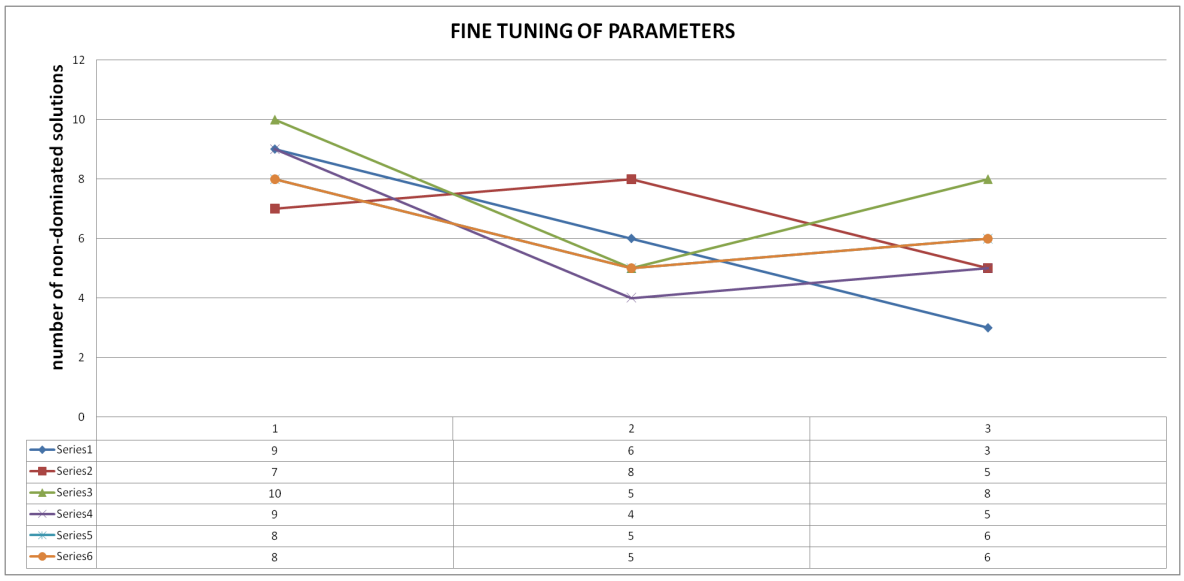
8.2.2 RESULTS FOR TLBO ALGORITHM:

X_new = X_i + r * [X_mean - (Tf * X_best)]
X_new = X_i + r * [X_i - (Tp * X_partner)]

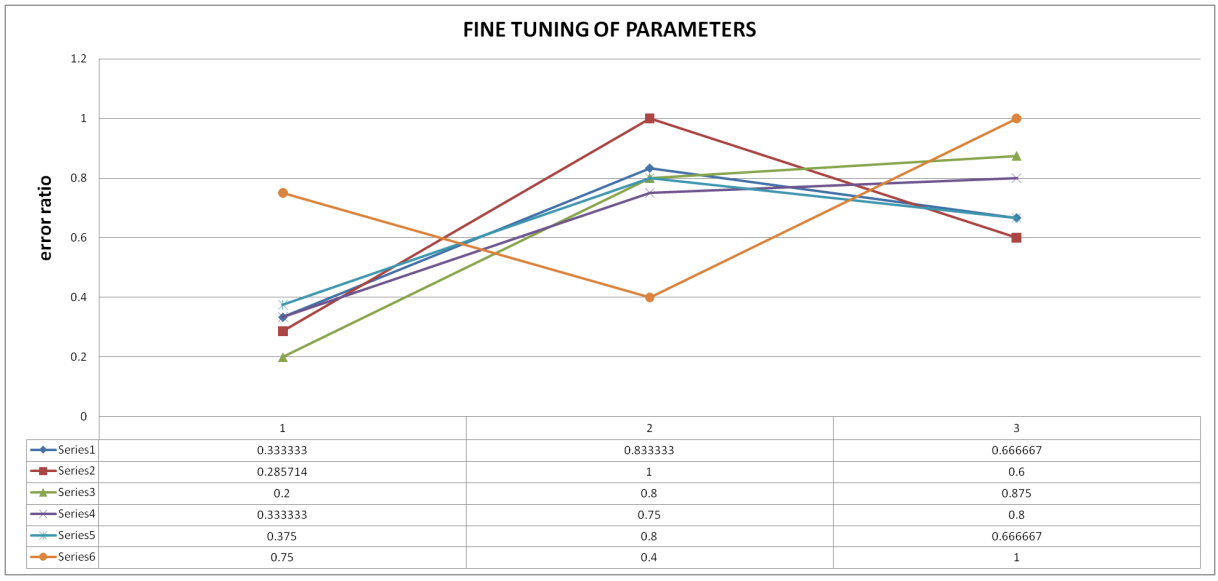
Where,
r = Coefficient that indicates the impact of the mean of the population and the teacher
Tf = Teaching coefficient that indicates the impact of teacher
Tp = Learning coefficient that indicates the impact of partner

- ★ **Series 1 -** r ∈ (0, 0.5), Tf+Tp <= 1.5
- ★ **Series 2 -** r ∈ (0.5, 1), Tf+Tp <= 1.5
- ★ **Series 3 -** Tf ∈ (0, 0.5), r+Tp <= 1.5
- ★ **Series 4 -** Tf ∈ (0.5, 1), r+Tp <= 1.5
- ★ **Series 5 -** Tp ∈ (0, 0.5), r+Tf <= 1.5
- ★ **Series 6 -** Tp ∈ (0.5, 1), r+Tf <= 1.5

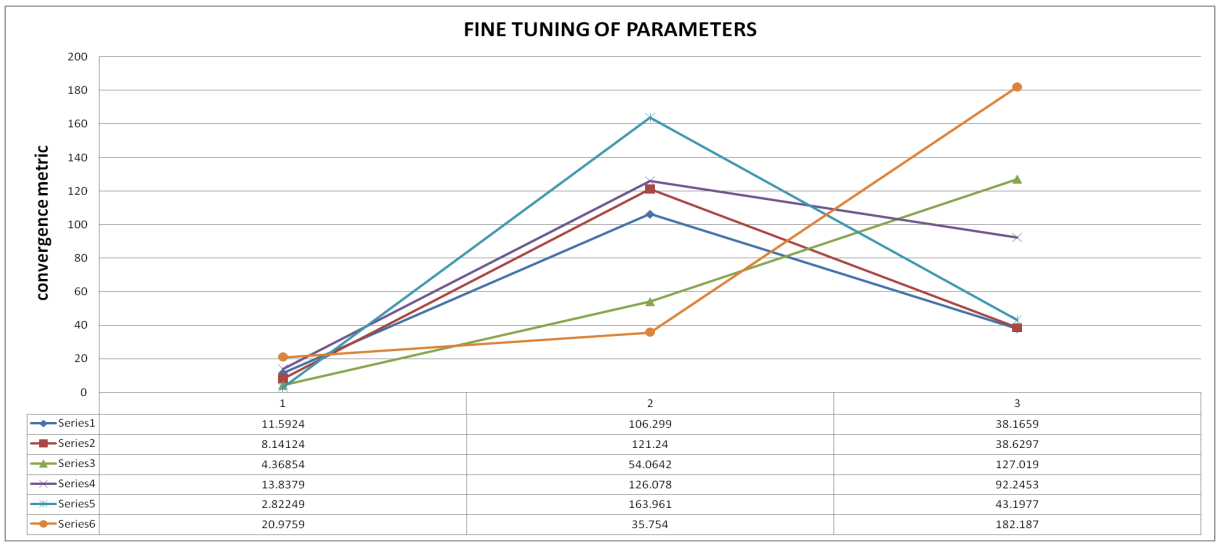
Following are the graphical plots for every performance indicator mentioned earlier VS the task dataset sizes- small, medium and large, represented by 1, 2 and 3 respectively for TLBO algorithm alone.



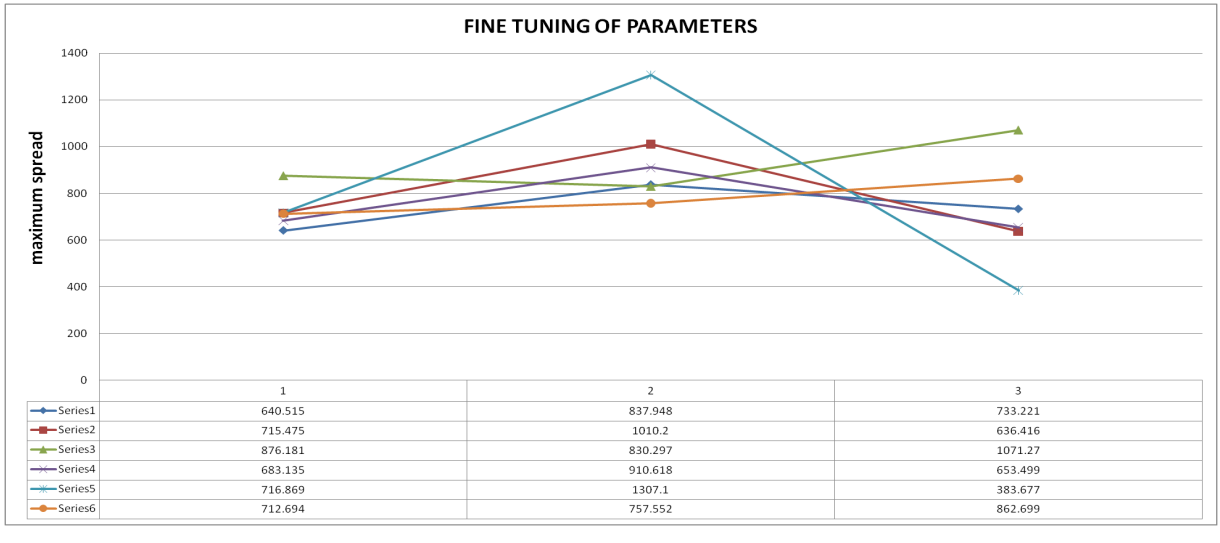
	1	2	3
Series1	9	6	3
Series2	7	8	5
Series3	10	5	8
Series4	9	4	5
Series5	8	5	6
Series6	8	5	6



	1	2	3
Series1	0.333333	0.833333	0.666667
Series2	0.285714	1	0.6
Series3	0.2	0.8	0.875
Series4	0.333333	0.75	0.8
Series5	0.375	0.8	0.666667
Series6	0.75	0.4	1



	1	2	3
Series1	11.5924	106.299	38.1659
Series2	8.14124	121.24	38.6297
Series3	4.36854	54.0642	127.019
Series4	13.8379	126.078	92.2453
Series5	2.82249	163.961	43.1977
Series6	20.9759	35.754	182.187



	1	2	3
Series1	640.515	837.948	733.221
Series2	715.475	1010.2	636.416
Series3	876.181	830.297	1071.27
Series4	683.135	910.618	653.499
Series5	716.869	1307.1	383.677
Series6	712.694	757.552	862.699

PROBLEM SIZE	PI 1	PI 2	PI 3	PI 4	PI 5	OVERALL
Small	series 3	series 3	series 5	series 5	series 3	**series 3**
Medium	series 2	series 6	series 6	series 2	series 5	**series 6**
Large	series 3	series 2	series 1	series 6	series 3	**series 3**

INFERENCE:

Based on the size of the task dataset, the results for the overall best performing "SERIES" range for TLBO algorithm is determined based on the consolidated PI results.

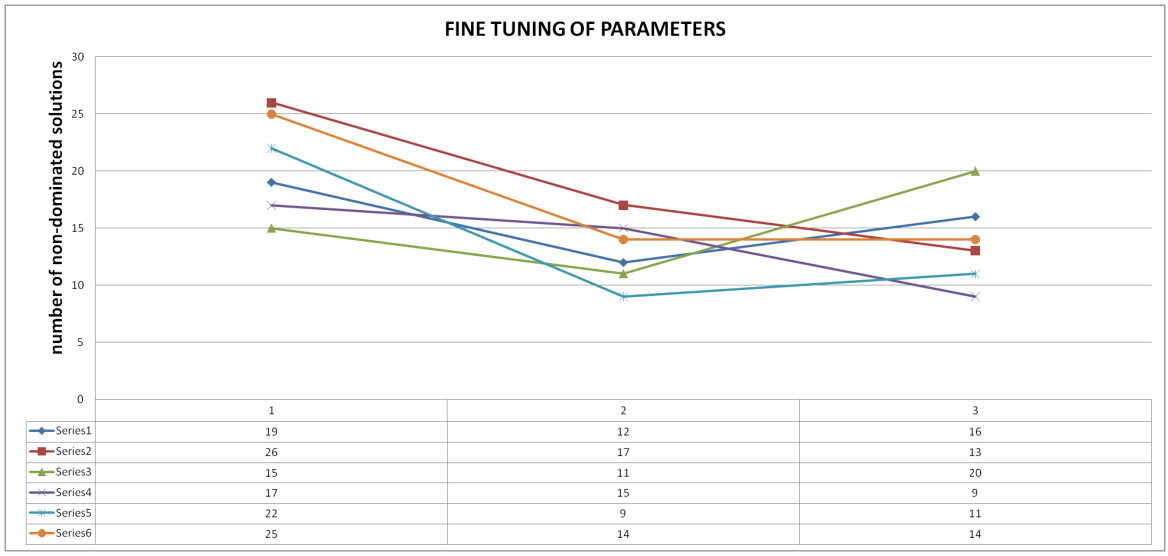
8.2.3 RESULTS FOR MBO ALGORITHM:

K - represents number of neighbourhood solutions of current leader (employed during LEADER IMPROVEMENT PHASE)

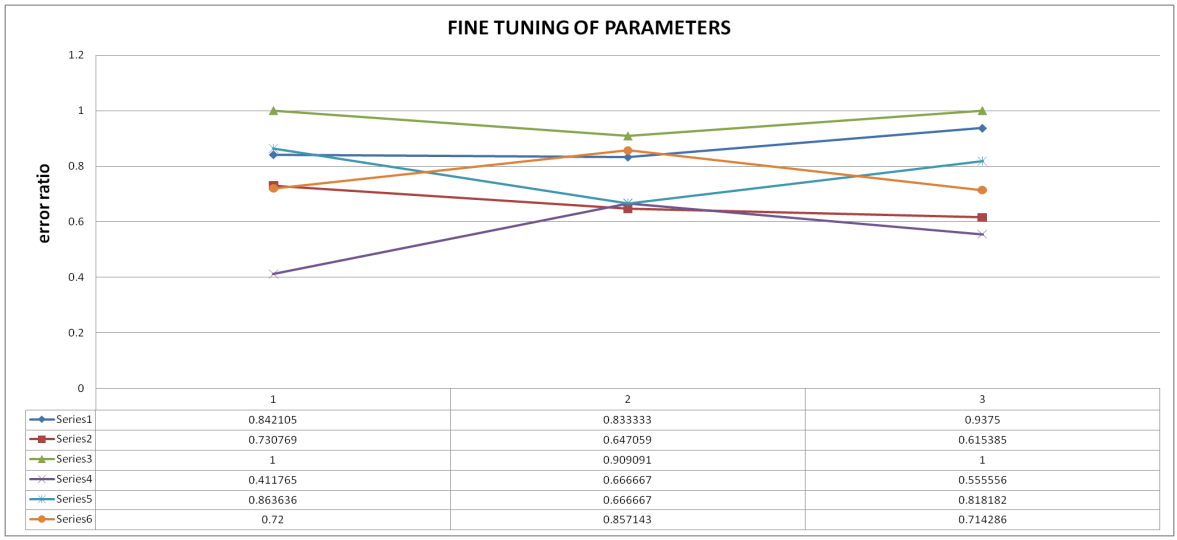
X - represents any number less than K. (whose expression together with K are employed during the BLOCK IMPROVEMENT PHASE)

- ★ **Series 1 -** K=2, X=1
- ★ **Series 2 -** K=3, X=1
- ★ **Series 3 -** K=5, X=2
- ★ **Series 4 -** K=5, X=3
- ★ **Series 5 -** K=7, X=3
- ★ **Series 6 -** K=10, X=5

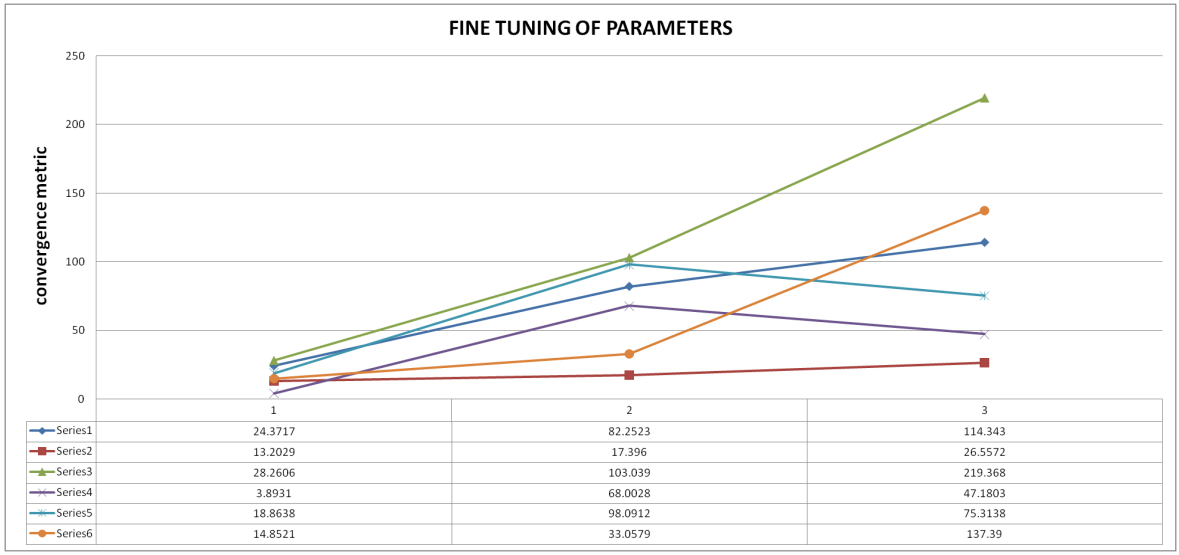
Following are the graphical plots for every performance indicator mentioned earlier VS the task dataset sizes- small, medium and large, represented by 1, 2 and 3 respectively for TLBO algorithm alone.



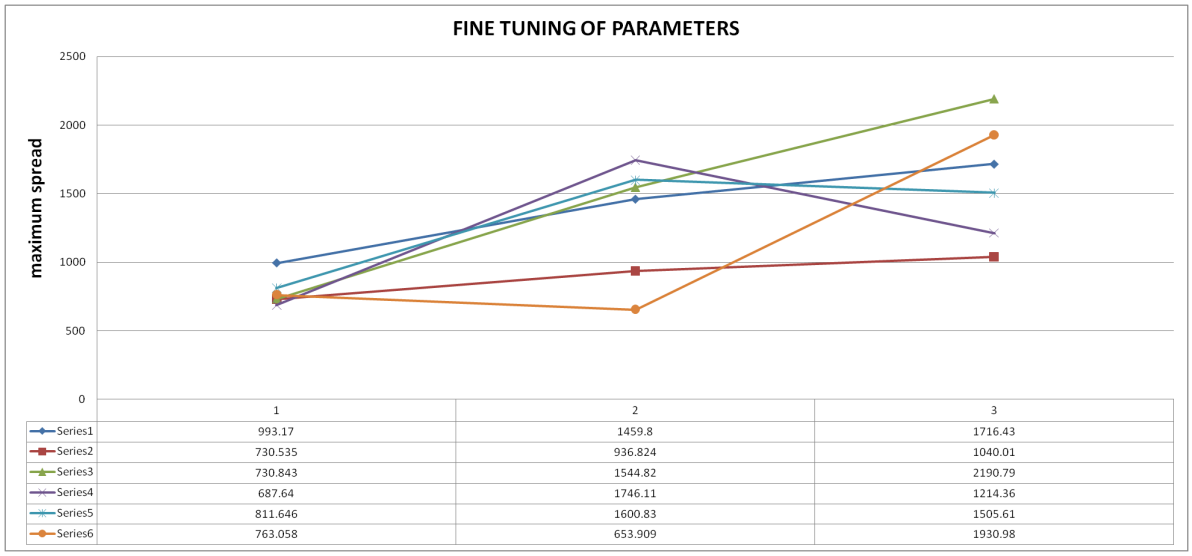
	1	2	3
Series1	19	12	16
Series2	26	17	13
Series3	15	11	20
Series4	17	15	9
Series5	22	9	11
Series6	25	14	14



	1	2	3
Series1	0.842105	0.833333	0.9375
Series2	0.730769	0.647059	0.615385
Series3	1	0.909091	1
Series4	0.411765	0.666667	0.555556
Series5	0.863636	0.666667	0.818182
Series6	0.72	0.857143	0.714286



	1	2	3
Series1	24.3717	82.2523	114.343
Series2	13.2029	17.396	26.5572
Series3	28.2606	103.039	219.368
Series4	3.8931	68.0028	47.1803
Series5	18.8638	98.0912	75.3138
Series6	14.8521	33.0579	137.39



PROBLEM SIZE	PI 1	PI 2	PI 3	PI 4	PI 5	OVERALL
Small	series 2	series 4	series 4	series 2	series 1	**series 4**
Medium	series 2	series 2	series 2	series 6	series 4	**series 2**
Large	series 3	series 4	series 2	series 6	series 3	**series 3**

INFERENCE:

Based on the size of the task dataset, the results for the overall best performing “SERIES” range for MBO algorithm is determined based on the consolidated PI results.

Based on the above results for individual algorithms, an overall result tabulation is shown.

ALGORITHM	PSO	TLBO	MBO	AOA
SMALL	c1 = [0 - 0.5] c2 + c3 <= 1.5	Tf = [0 - 0.5] r + Tp <= 1.5	K = 5 X = 3	-------
MEDIUM	c3 = [0.5 - 1] c1 + c2 <= 1.5	Tp = [0.5 - 1] r + Tf <= 1.5	K = 3 X = 1	-------
LARGE	c2 = [0.5 - 1] c1 + c3 <= 1.5	Tf = [0 - 0.5] r + Tp <= 1.5	K = 5 X = 2	-------

These results can be inferred by researchers working on the improvement of the same algorithms as above, saving them much computational efforts in fine tuning the parameters of the above algorithms, according to the size of the problem considered by them.

8.3 CONSOLIDATED GRAPHICAL COMPARISON OF ALL STANDARD vs HYBRID ALGORITHMS:

- ❖ After adding the scout phase for the four algorithms developed, we now have to compare all the 8 algorithms.
- ❖ To make the comparison easier, we segregated the data as small, medium and large as mentioned in earlier slides.
- ❖ A total of 32 datasets is generated and used to compare these algorithms using the 5 performance indicators.
- ❖ The description of each data set is explained in upcoming slides.

The base four algorithms,

- ➔ **PSO** (particle swarm optimisation) - **ALGO 1**
- ➔ **TLBO** (teaching learning based optimisation) - **ALGO 2**
- ➔ **MBO** (migrating bird optimisation) - **ALGO 3**
- ➔ **AOA** (archimedes optimisation algorithm) - **ALGO 4**

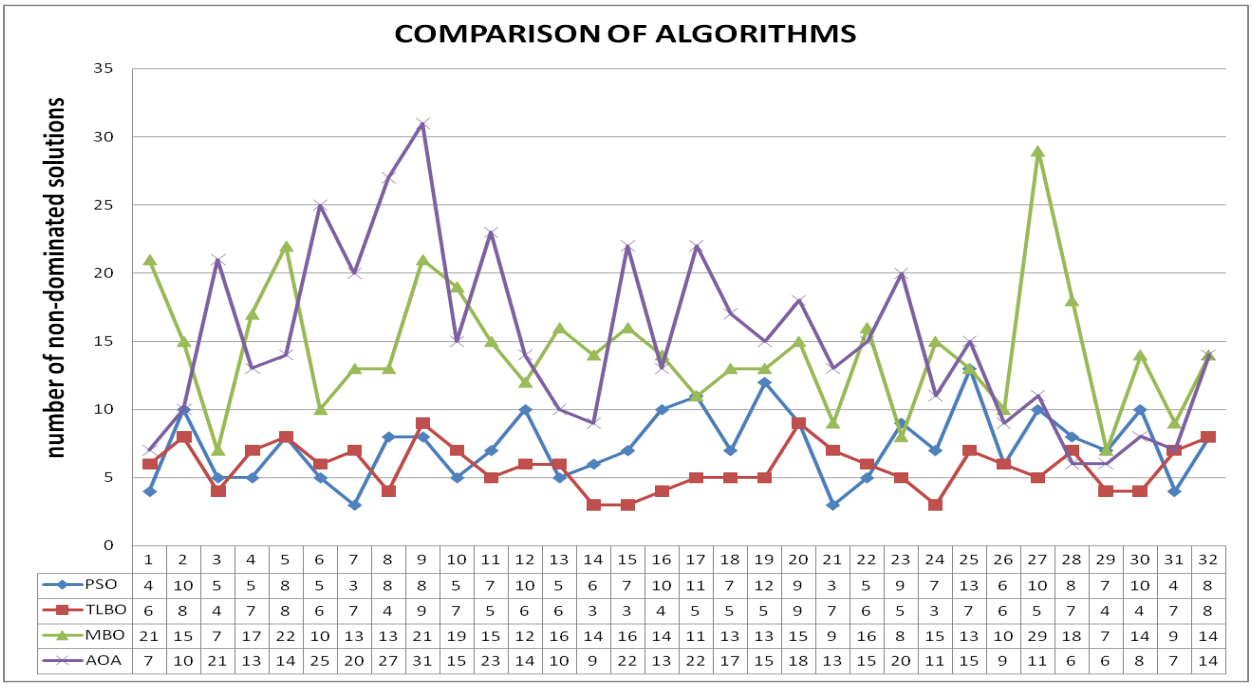
And the variants,

- ➔ **PSO + ABC** (particle swarm optimisation with scout phase) - **ALGO 5**
- ➔ **TLBO + ABC** (teaching learning based optimisation with scout phase) - **ALGO 6**
- ➔ **MBO + ABC** (migrating bird optimisation with scout phase) - **ALGO 7**

➔ **AOA + ABC** (archimedes optimisation algorithm with scout phase) - **ALGO 8**

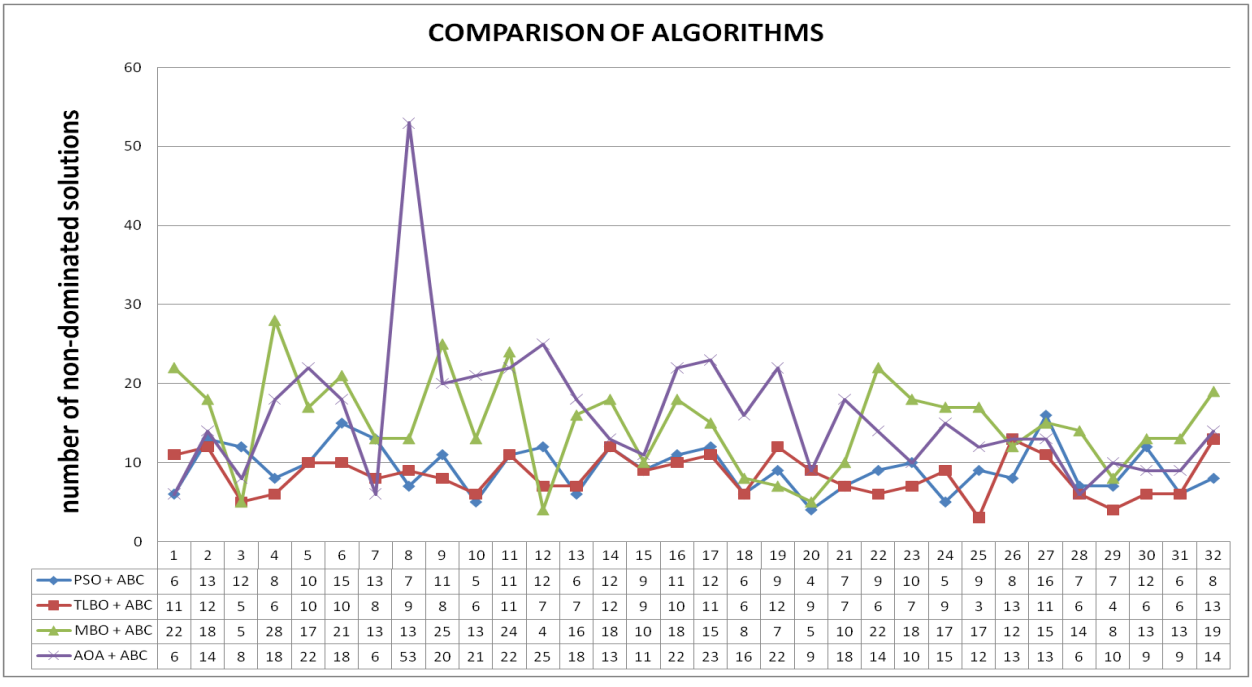
These are plotted in separate graphs to make it easier for the user to visualise the graphs. The results obtained are as follows:

☐ **PI-1 for all 32 datasets for all four standard algorithms:**



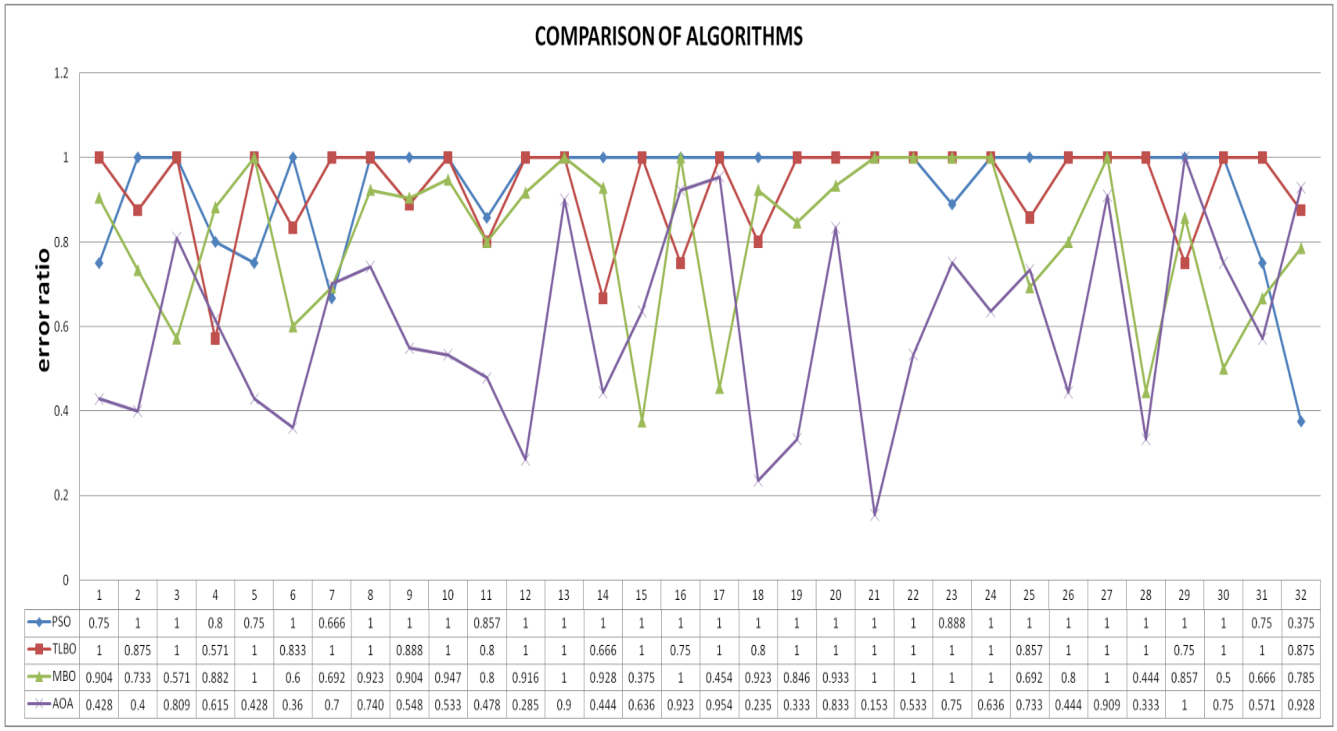
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
PSO	4	10	5	5	8	5	3	8	8	5	7	10	5	6	7	10	11	7	12	9	3	5	9	7	13	6	10	8	7	10	4	8
TLBO	6	8	4	7	8	6	7	4	9	7	5	6	6	3	3	4	5	5	5	9	7	6	5	3	7	6	5	7	4	4	7	8
MBO	21	15	7	17	22	10	13	13	21	19	15	12	16	14	16	14	11	13	13	15	9	16	8	15	13	10	29	18	7	14	9	14
AOA	7	10	21	13	14	25	20	27	31	15	23	14	10	9	22	13	22	17	15	18	13	15	20	11	15	9	11	6	6	8	7	14

☐ **PI-1 for all 32 datasets for all four hybridised algorithms:**



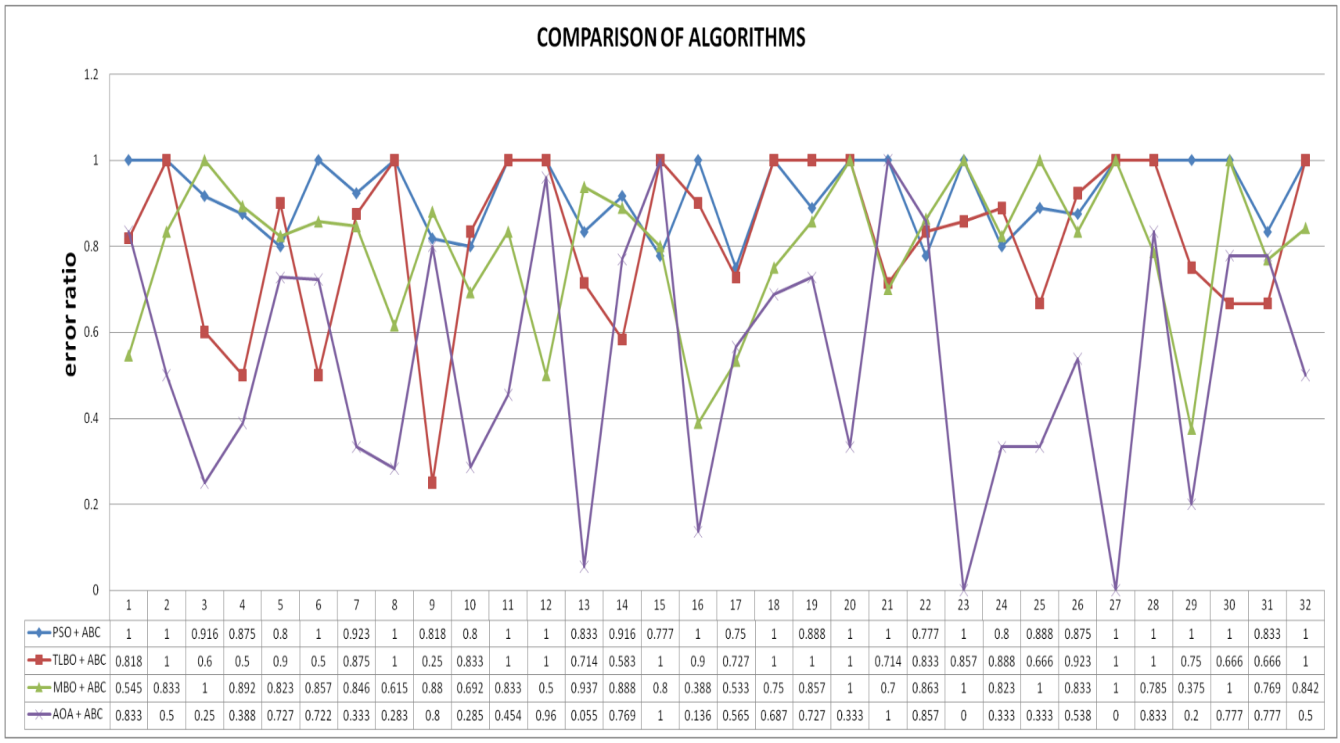
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
PSO + ABC	6	13	12	8	10	15	13	7	11	5	11	12	6	12	9	11	12	6	9	4	7	9	10	5	9	8	16	7	7	12	6	8
TLBO + ABC	11	12	5	6	10	10	8	9	8	6	11	7	7	12	9	10	11	6	12	9	7	6	7	9	3	13	11	6	4	6	6	13
MBO + ABC	22	18	5	28	17	21	13	13	25	13	24	4	16	18	10	18	15	8	7	5	10	22	18	17	17	12	15	14	8	13	13	19
AOA + ABC	6	14	8	18	22	18	6	53	20	21	22	25	18	13	11	22	23	16	22	9	18	14	10	15	12	13	13	6	10	9	9	14

☐ **PI-2 for all 32 datasets for all four standard algorithms:**



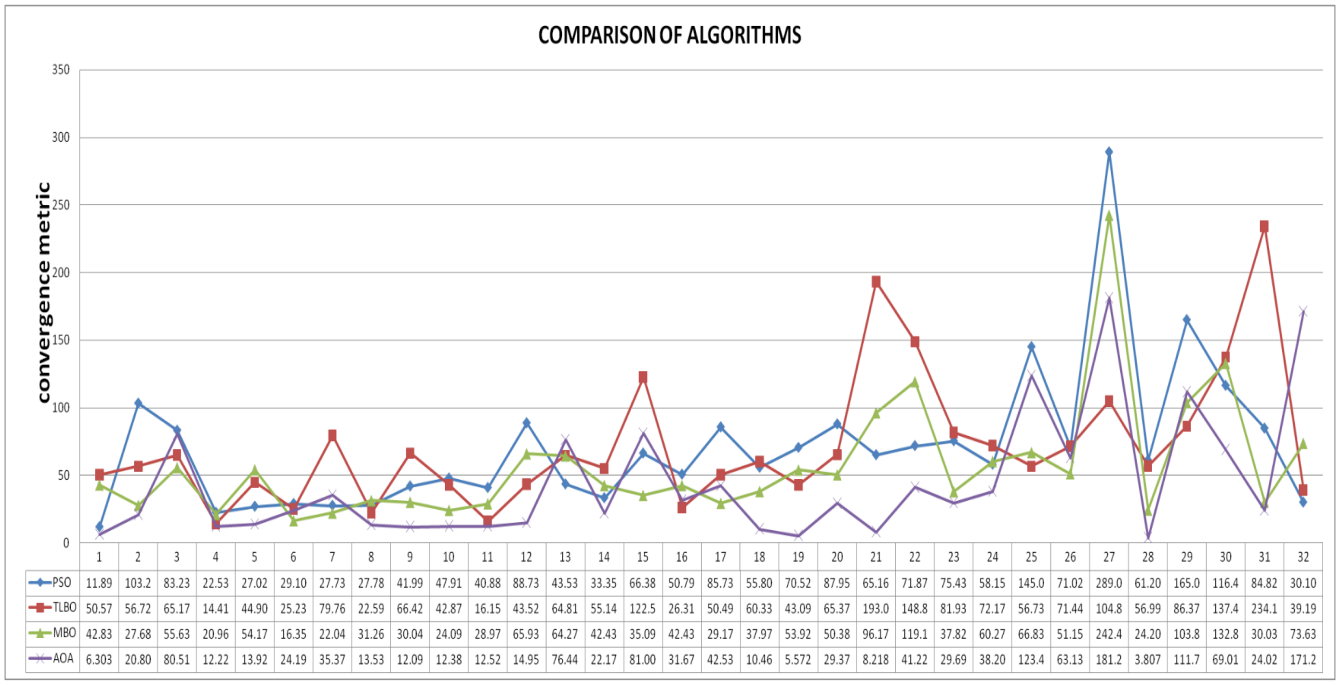
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
PSO	0.75	1	1	0.8	0.75	1	0.666	1	1	1	0.857	1	1	1	1	1	1	1	1	1	1	1	0.888	1	1	1	1	1	1	1	0.75	0.375
TLBO	1	0.875	1	0.571	1	0.833	1	1	0.888	1	0.8	1	1	0.666	1	0.75	1	0.8	1	1	1	1	1	1	0.857	1	1	1	0.75	1	1	0.875
MBO	0.904	0.733	0.571	0.882	1	0.6	0.692	0.923	0.904	0.947	0.8	0.916	1	0.928	0.375	1	0.454	0.923	0.846	0.933	1	1	1	1	0.692	0.8	1	0.444	0.857	0.5	0.666	0.785
AOA	0.428	0.4	0.809	0.615	0.428	0.36	0.7	0.740	0.548	0.533	0.478	0.285	0.9	0.444	0.636	0.923	0.954	0.235	0.333	0.833	0.153	0.533	0.75	0.636	0.733	0.444	0.909	0.333	1	0.75	0.571	0.928

☐ **PI-2 for all 32 datasets for all four hybridised algorithms:**



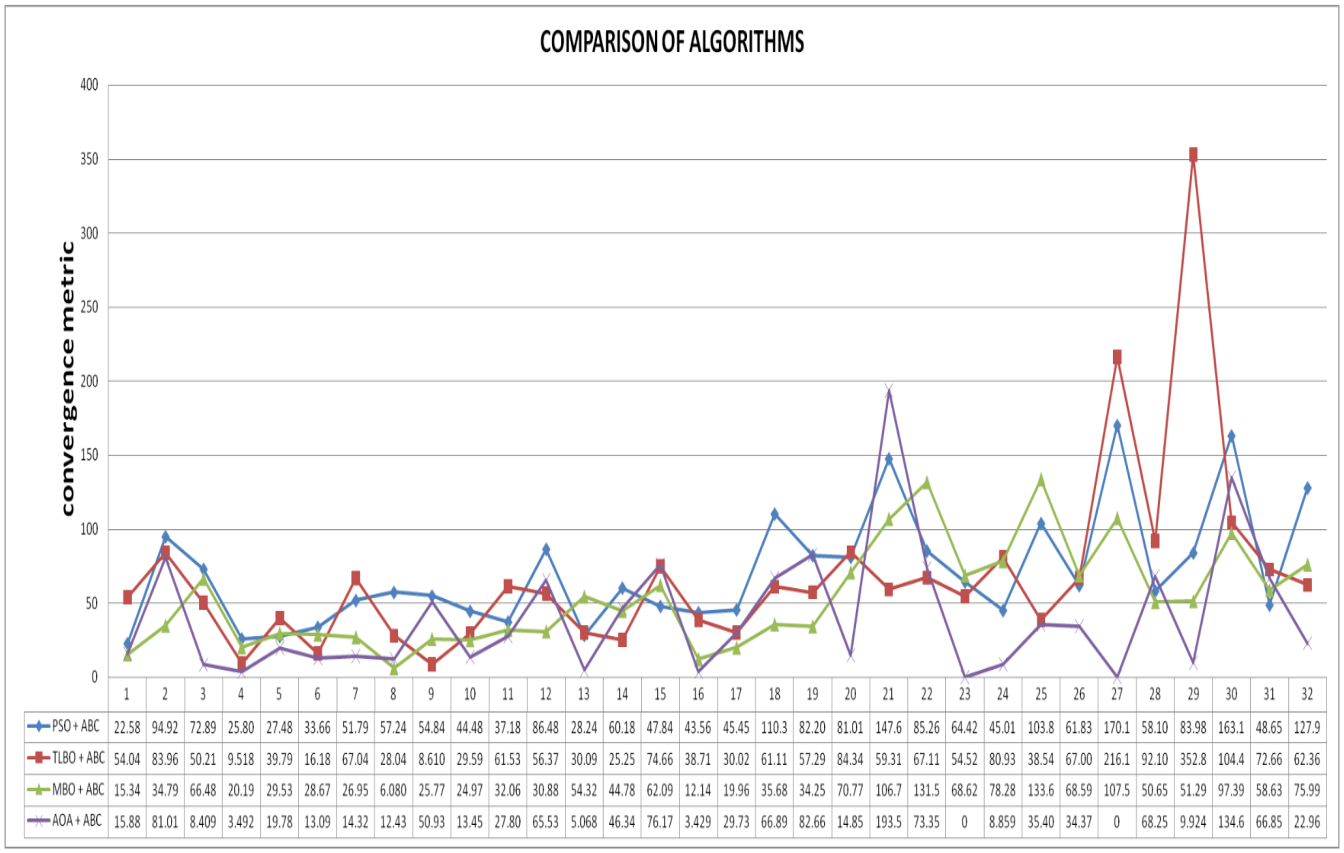
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
PSO + ABC	1	1	0.916	0.875	0.8	1	0.923	1	0.818	0.8	1	1	0.833	0.916	0.777	1	0.75	1	0.888	1	1	0.777	1	0.8	0.888	0.875	1	1	1	1	0.833	1
TLBO + ABC	0.818	1	0.6	0.5	0.9	0.5	0.875	1	0.25	0.833	1	1	0.714	0.583	1	0.9	0.727	1	1	1	0.714	0.833	0.857	0.888	0.666	0.923	1	1	0.75	0.666	0.666	1
MBO + ABC	0.545	0.833	1	0.892	0.823	0.857	0.846	0.615	0.88	0.692	0.833	0.5	0.937	0.888	0.8	0.388	0.533	0.75	0.857	1	0.7	0.863	1	0.823	1	0.833	1	0.785	0.375	1	0.769	0.842
AOA + ABC	0.833	0.5	0.25	0.388	0.727	0.722	0.333	0.283	0.8	0.285	0.454	0.96	0.055	0.769	1	0.136	0.565	0.687	0.727	0.333	1	0.857	0	0.333	0.333	0.538	0	0.833	0.2	0.777	0.777	0.5

☐ **PI-3 for all 32 datasets for all four standard algorithms:**



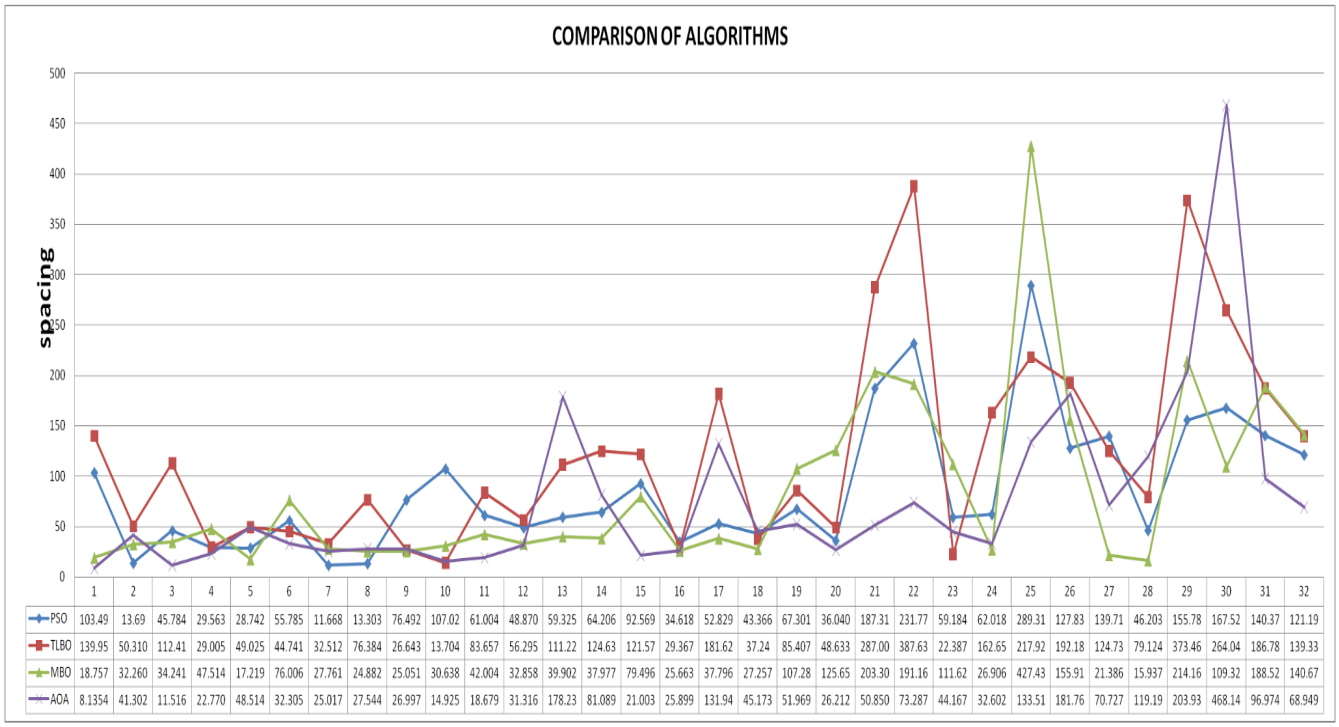
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
PSO	11.89	103.2	83.23	22.53	27.02	29.10	27.73	27.78	41.99	47.91	40.88	88.73	43.53	33.35	66.38	50.79	85.73	55.80	70.52	87.95	65.16	71.87	75.43	58.15	145.0	71.02	289.0	61.20	165.0	116.4	84.82	30.10
TLBO	50.57	56.72	65.17	14.41	44.90	25.23	79.76	22.59	66.42	42.87	16.15	43.52	64.81	55.14	122.5	26.31	50.49	60.33	43.09	65.37	193.0	148.8	81.93	72.17	56.73	71.44	104.8	56.99	86.37	137.4	234.1	39.19
MBO	42.83	27.68	55.63	20.96	54.17	16.35	22.04	31.26	30.04	24.09	28.97	65.93	64.27	42.43	35.09	42.43	29.17	37.97	53.92	50.38	96.17	119.1	37.82	60.27	66.83	51.15	242.4	24.20	103.8	132.8	30.03	73.63
AOA	6.303	20.80	80.51	12.22	13.92	24.19	35.37	13.53	12.09	12.38	12.52	14.95	76.44	22.17	81.00	31.67	42.53	10.46	5.572	29.37	8.218	41.22	29.69	38.20	123.4	63.13	181.2	3.807	111.7	69.01	24.02	171.2

☐ **PI-3 for all 32 datasets for all four hybridised algorithms:**



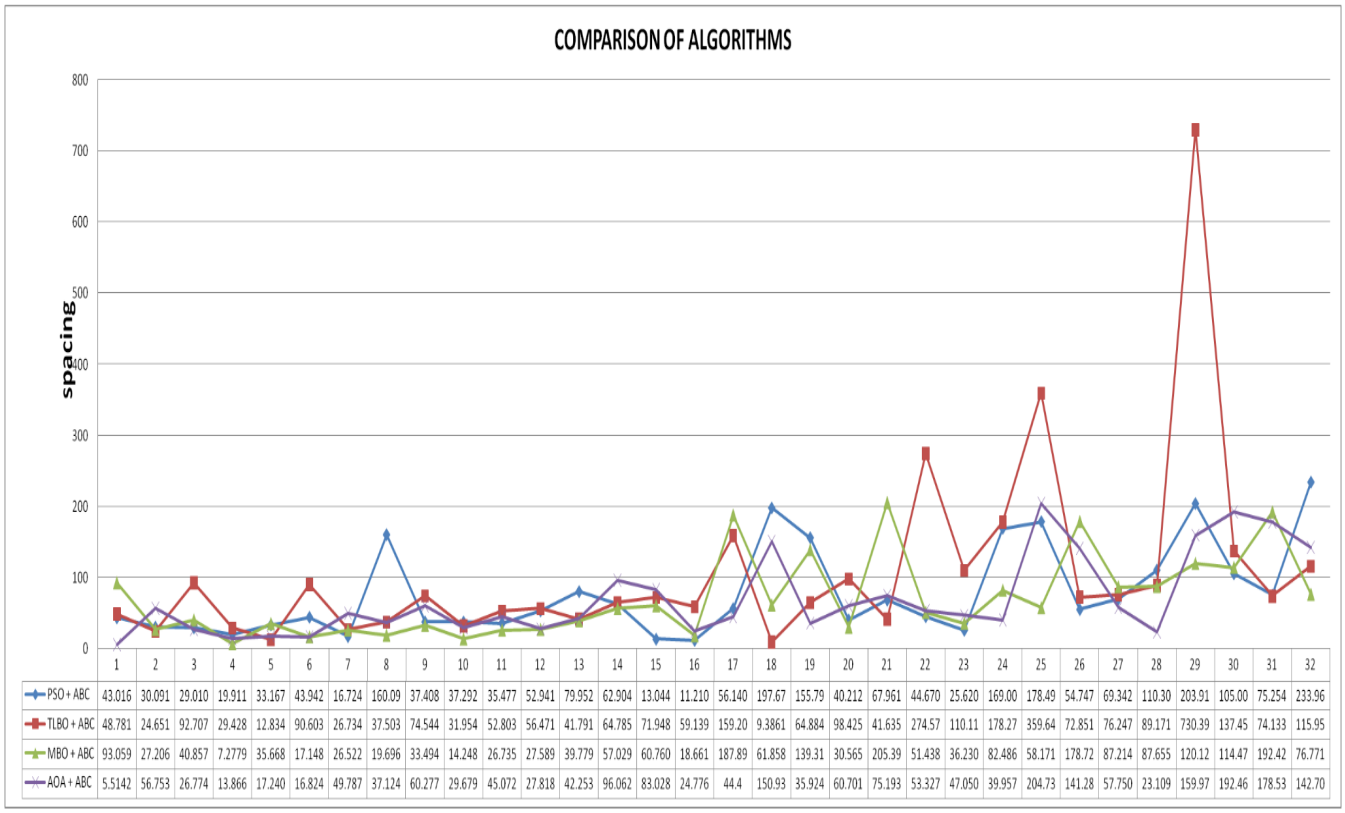
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
PSO + ABC	22.58	94.92	72.89	25.80	27.48	33.66	51.79	57.24	54.84	44.48	37.18	86.48	28.24	60.18	47.84	43.56	45.45	110.3	82.20	81.01	147.6	85.26	64.42	45.01	103.8	61.83	170.1	58.10	83.98	163.1	48.65	127.9
TLBO + ABC	54.04	83.96	50.21	9.518	39.79	16.18	67.04	28.04	8.610	29.59	61.53	56.37	30.09	25.25	74.66	38.71	30.02	61.11	57.29	84.34	59.31	67.11	54.52	80.93	38.54	67.00	216.1	92.10	352.8	104.4	72.66	62.36
MBO + ABC	15.34	34.79	66.48	20.19	29.53	28.67	26.95	6.080	25.77	24.97	32.06	30.88	54.32	44.78	62.09	12.14	19.96	35.68	34.25	70.77	106.7	131.5	68.62	78.28	133.6	68.59	107.5	50.65	51.29	97.39	58.63	75.99
AOA + ABC	15.88	81.01	8.409	3.492	19.78	13.09	14.32	12.43	50.93	13.45	27.80	65.53	5.068	46.34	76.17	3.429	29.73	66.89	82.66	14.85	193.5	73.35	0	8.859	35.40	34.37	0	68.25	9.924	134.6	66.85	22.96

☐ **PI-4 for all 32 datasets for all four standard algorithms:**



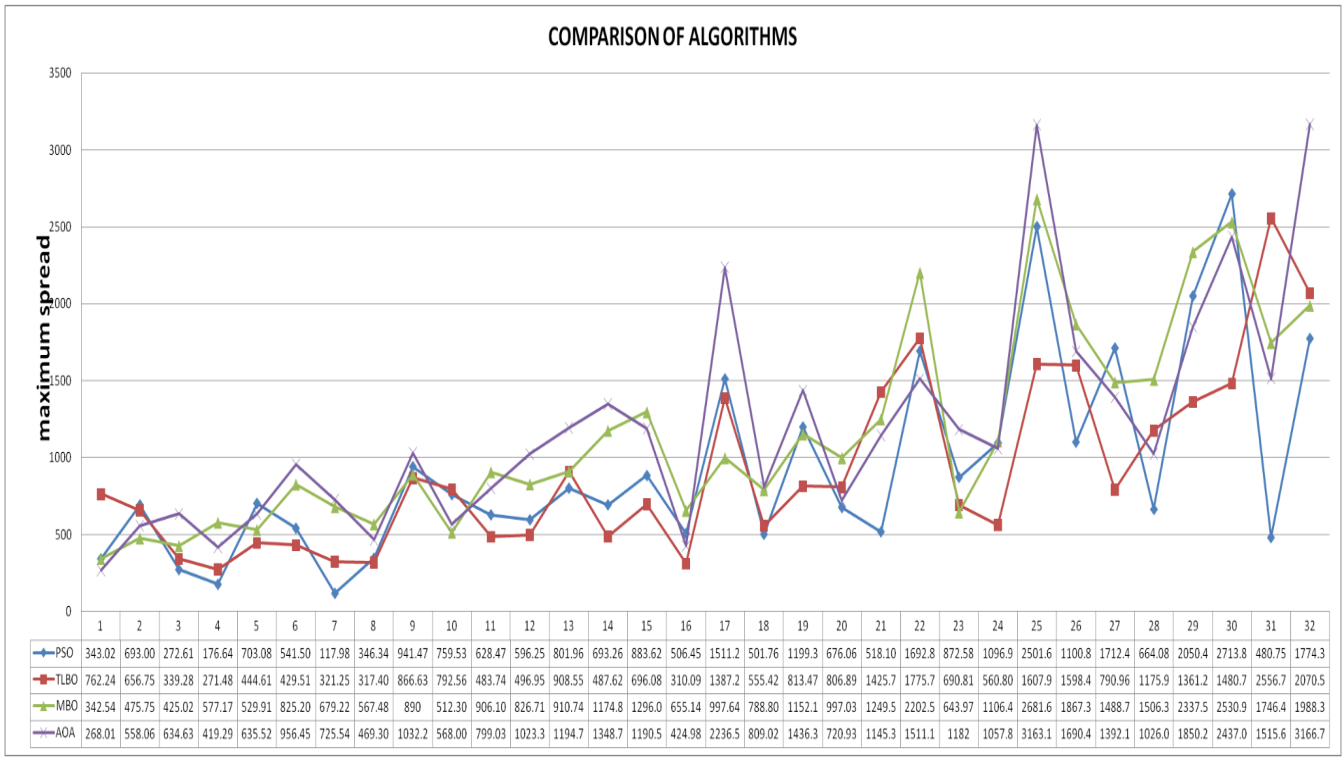
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
PSO	103.49	13.69	45.784	29.563	28.742	55.785	11.668	13.303	76.492	107.02	61.004	48.870	59.325	64.206	92.569	34.618	52.829	43.366	67.301	36.040	187.31	231.77	59.184	62.018	289.31	127.83	139.71	46.203	155.78	167.52	140.37	121.19
TLBO	139.95	50.310	112.41	29.005	49.025	44.741	32.512	76.384	26.643	13.704	83.657	56.295	111.22	124.63	121.57	29.367	181.62	37.24	85.407	48.633	287.00	387.63	22.387	162.65	217.92	192.18	124.73	79.124	373.46	264.04	186.78	139.33
MBO	18.757	32.260	34.241	47.514	17.219	76.006	27.761	24.882	25.051	30.638	42.004	32.858	39.902	37.977	79.496	25.663	37.796	27.257	107.28	125.65	203.30	191.16	111.62	26.906	427.43	155.91	21.386	15.937	214.16	109.32	188.52	140.67
AOA	8.1354	41.302	11.516	22.770	48.514	32.305	25.017	27.544	26.997	14.925	18.679	31.316	178.23	81.089	21.003	25.899	131.94	45.173	51.969	26.212	50.850	73.287	44.167	32.602	133.51	181.76	70.727	119.19	203.93	468.14	96.974	68.949

☐ **PI-4 for all 32 datasets for all four hybridised algorithms:**



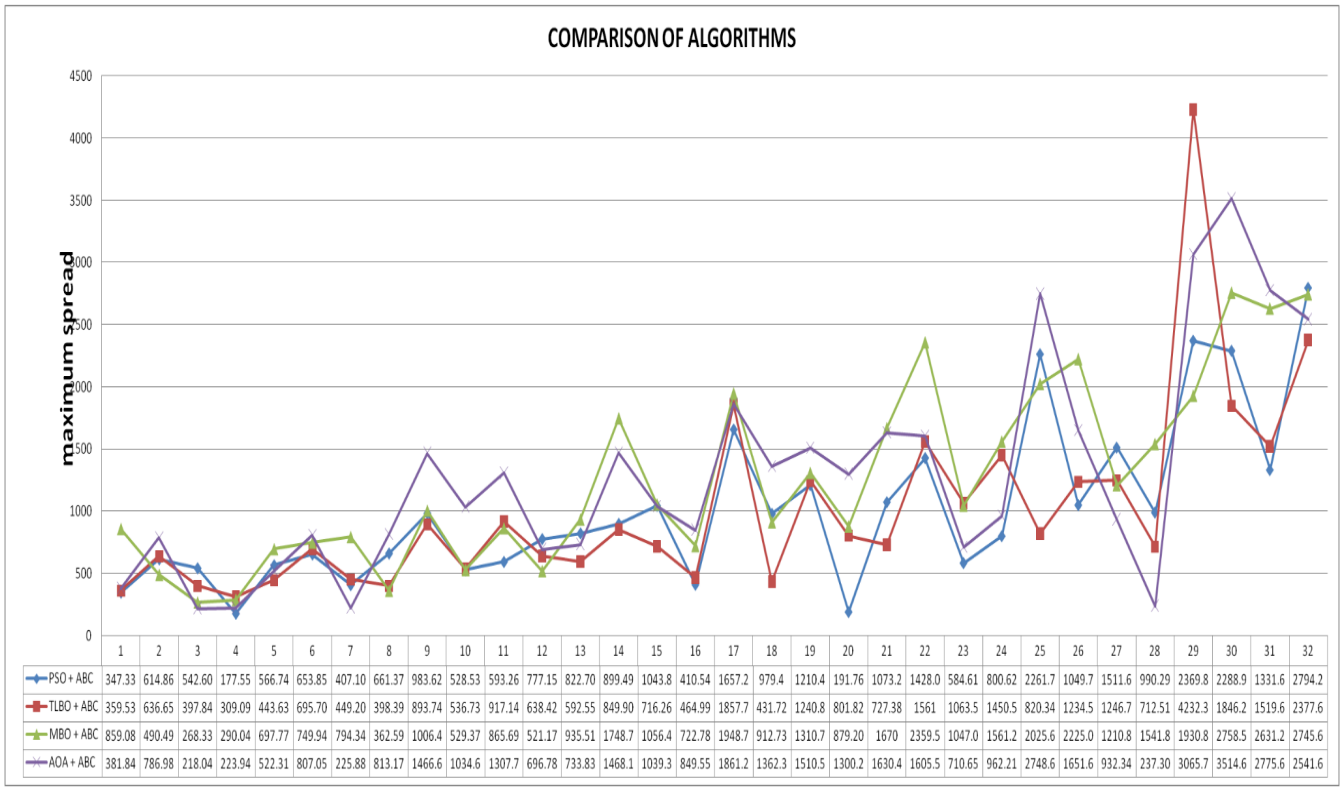
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
PSO+ABC	43.016	30.091	29.010	19.911	33.167	43.942	16.724	160.09	37.408	37.292	35.477	52.941	79.952	62.904	13.044	11.210	56.140	197.67	155.79	40.212	67.961	44.670	25.620	169.00	178.49	54.747	69.342	110.30	203.91	105.00	75.254	233.96
TLBO+ABC	48.781	24.651	92.707	29.428	12.834	90.603	26.734	37.503	74.544	31.954	52.803	56.471	41.791	64.785	71.948	59.139	159.20	9.3861	64.884	98.425	41.635	274.57	110.11	178.27	359.64	72.851	76.247	89.171	730.39	137.45	74.133	115.95
MBO+ABC	93.059	27.206	40.857	7.2779	35.668	17.148	26.522	19.696	33.494	14.248	26.735	27.589	39.779	57.029	60.760	18.661	187.89	61.858	139.31	30.565	205.39	51.438	36.230	82.486	58.171	178.72	87.214	87.655	120.12	114.47	192.42	76.771
AOA+ABC	5.5142	56.753	26.774	13.866	17.240	16.824	49.787	37.124	60.277	29.679	45.072	27.818	42.253	96.062	83.028	24.776	44.4	150.93	35.924	60.701	75.193	53.327	47.050	39.957	204.73	141.28	57.750	23.109	159.97	192.46	178.53	142.70

☐ **PI-5 for all 32 datasets for all four standard algorithms:**



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
PSO	343.02	693.00	272.61	176.64	703.08	541.50	117.98	346.34	941.47	759.53	628.47	596.25	801.96	693.26	883.62	506.45	1511.2	501.76	1199.3	676.06	518.10	1692.8	872.58	1096.9	2501.6	1100.8	1712.4	664.08	2050.4	2713.8	480.75	1774.3
TLBO	762.24	656.75	339.28	271.48	444.61	429.51	321.25	317.40	866.63	792.56	483.74	496.95	908.55	487.62	696.08	310.09	1387.2	555.42	813.47	806.89	1425.7	1775.7	690.81	560.80	1607.9	1598.4	790.96	1175.9	1361.2	1480.7	2556.7	2070.5
MBO	342.54	475.75	425.02	577.17	529.91	825.20	679.22	567.48	890	512.30	906.10	826.71	910.74	1174.8	1296.0	655.14	997.64	788.80	1152.1	997.03	1249.5	2202.5	643.97	1106.4	2681.6	1867.3	1488.7	1506.3	2337.5	2530.9	1746.4	1988.3
AOA	268.01	558.06	634.63	419.29	635.52	956.45	725.54	469.30	1032.2	568.00	799.03	1023.3	1194.7	1348.7	1190.5	424.98	2236.5	809.02	1436.3	720.93	1145.3	1511.1	1182	1057.8	3163.1	1690.4	1392.1	1026.0	1850.2	2437.0	1515.6	3166.7

☐ **PI-5 for all 32 datasets for all four hybridised algorithms:**



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
PSO + ABC	347.33	614.86	542.60	177.55	566.74	653.85	407.10	661.37	983.62	528.53	593.26	777.15	822.70	899.49	1043.8	410.54	1657.2	979.4	1210.4	191.76	1073.2	1428.0	584.61	800.62	2261.7	1049.7	1511.6	990.29	2369.8	2288.9	1331.6	2794.2
TLBO + ABC	359.53	636.65	397.84	309.09	443.63	695.70	449.20	398.39	893.74	536.73	917.14	638.42	592.55	849.90	716.26	464.99	1857.7	431.72	1240.8	801.82	727.38	1561	1063.5	1450.5	820.34	1234.5	1246.7	712.51	4232.3	1846.2	1519.6	2377.6
MBO + ABC	859.08	490.49	268.33	290.04	697.77	749.94	794.34	362.59	1006.4	529.37	865.69	521.17	935.51	1748.7	1056.4	722.78	1948.7	912.73	1310.7	879.20	1670	2359.5	1047.0	1561.2	2025.6	2225.0	1210.8	1541.8	1930.8	2758.5	2631.2	2745.6
AOA + ABC	381.84	786.98	218.04	223.94	522.31	807.05	225.88	813.17	1466.6	1034.6	1307.7	696.78	733.83	1468.1	1039.3	849.55	1861.2	1362.3	1510.5	1300.2	1630.4	1605.5	710.65	962.21	2748.6	1651.6	932.34	237.30	3065.7	3514.6	2775.6	2541.6

8.4 CONSOLIDATED NUMERICAL COMPARISON OF ALL STANDARD vs HYBRID ALGORITHMS:

DATA SET no.	No. of tasks	No. of work stations	PROBLEM SIZE	PI 1	PI 2	PI 3	PI 4	PI 5	OVERALL
1	25	3	**SMALL**	6	7	7	8	6	**6, 7**
2	25	4	**SMALL**	6	7	7	1	8	**7**
3	25	6	**SMALL**	7	8	8	7	7	**7**
4	25	9	**SMALL**	6	8	8	6	5	**6, 8**
5	35	4	**SMALL**	8	7	7	4	1	**7**
6	35	5	**SMALL**	7	7	8	8	7	**7**
7	35	7	**SMALL**	7	8	8	1	6	**8**
8	35	12	**SMALL**	8	8	6	1	8	**8**
9	53	5	**MEDIUM**	7	4	4	5	8	**4**
10	53	7	**MEDIUM**	8	8	7	3	8	**8**
11	53	10	**MEDIUM**	6	8	7	7	8	**7, 8**
12	53	14	**MEDIUM**	8	7	7	6	7	**7**
13	70	7	**MEDIUM**	8	8	8	6	7	**8**
14	70	10	**MEDIUM**	6	7	7	5	6	**6, 7**

15	70	14	**MEDIUM**	7	5	5	2	5	**5**
16	70	19	**MEDIUM**	8	8	8	2	8	**8**
17	89	8	**MEDIUM**	8	5	6	5	7	**5**
18	89	12	**MEDIUM**	7	7	7	4	8	**7**
19	89	16	**MEDIUM**	8	7	7	8	8	**8**
20	89	21	**MEDIUM**	7	8	8	7	8	**8**

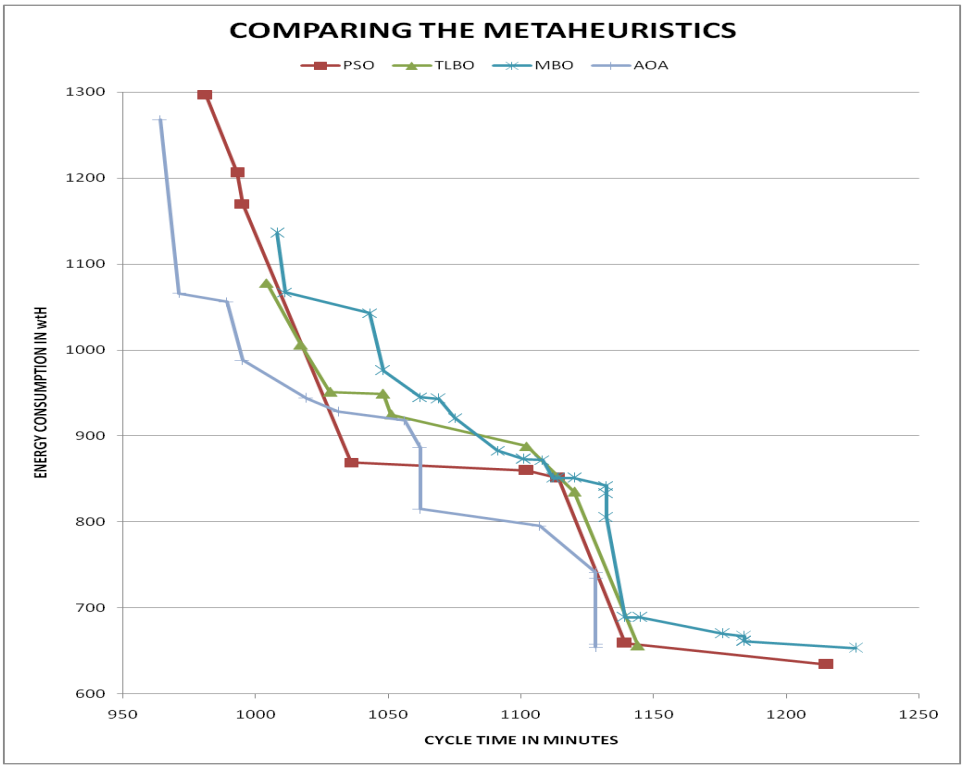
21	111	9	**LARGE**	8	7	7	4	6	**7**
22	111	13	**LARGE**	6	7	7	2	6	**6, 7**
23	111	17	**LARGE**	7	8	8	3	7	**7, 8**
24	111	22	**LARGE**	6	8	8	5	6	**6, 8**
25	148	10	**LARGE**	6	8	8	6	7	**6, 8**
26	148	14	**LARGE**	8	7	8	2	6	**8**
27	148	21	**LARGE**	5	8	8	5	1	**5, 8**

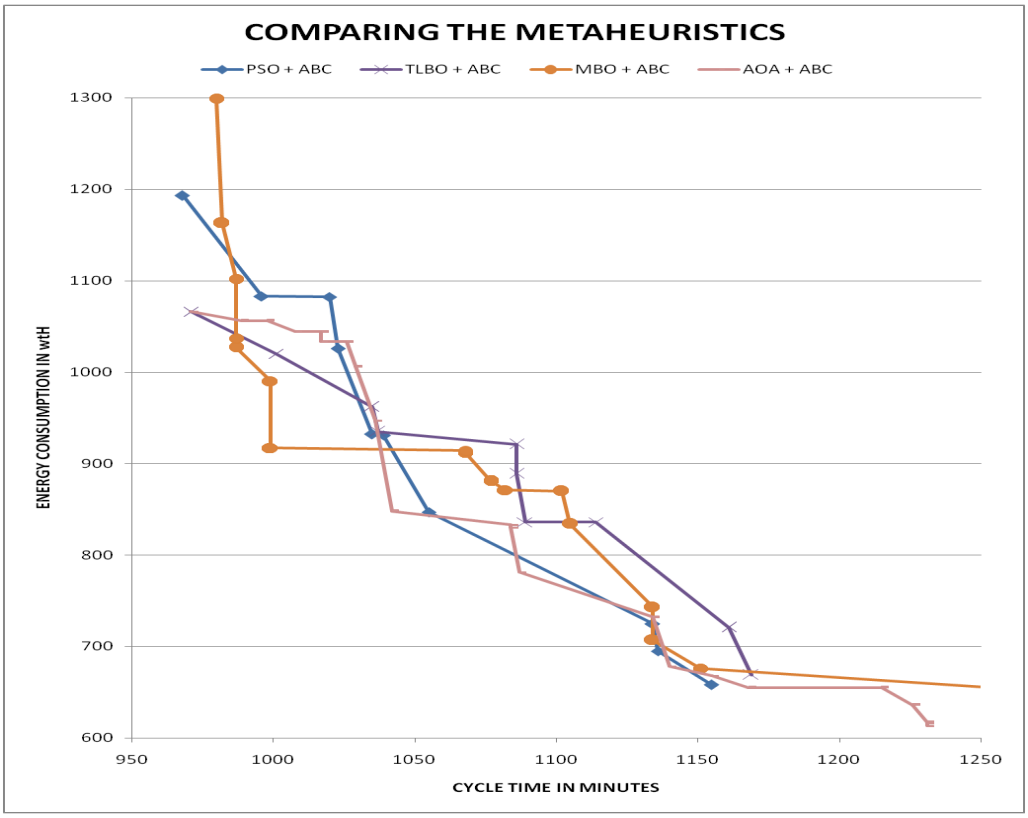
28	148	29	**LARGE**	5	7	7	5	6	**5, 7**
29	297	19	**LARGE**	8	8	8	6	4	**8**
30	297	29	**LARGE**	5	5	7	2	8	**5**
31	297	38	**LARGE**	6	7	7	4	8	**7**

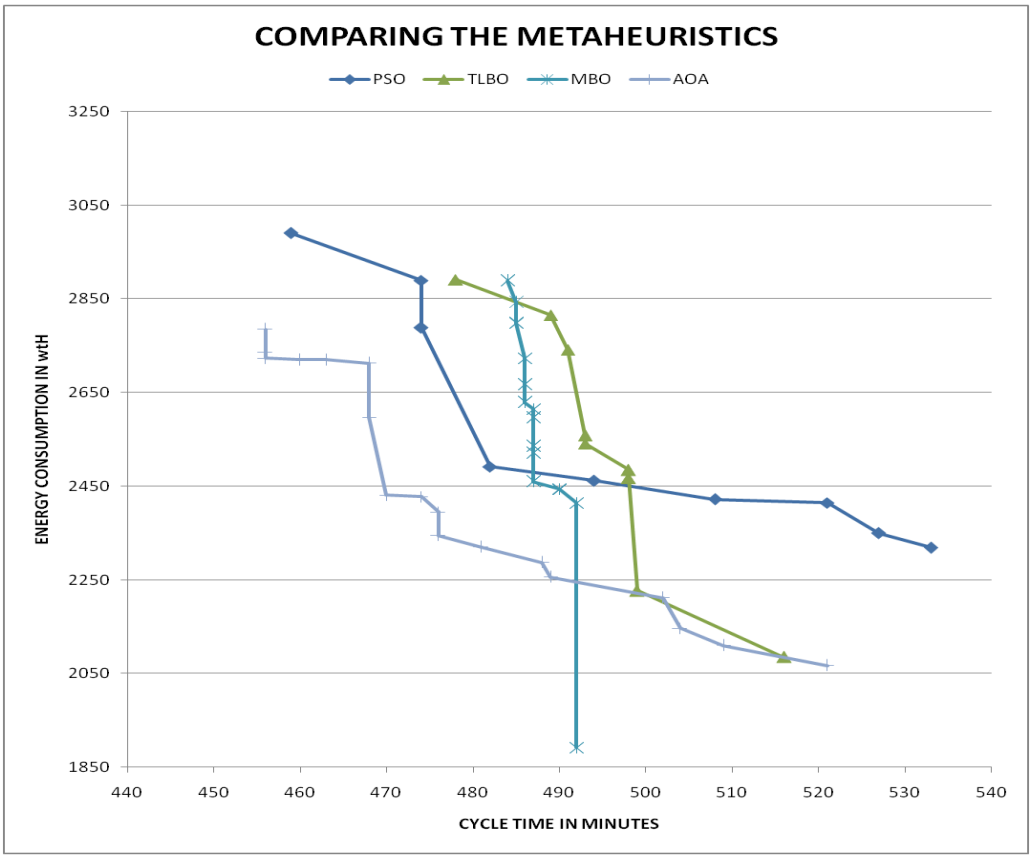
32	297	50	LARGE	6	1	8	7	7	7

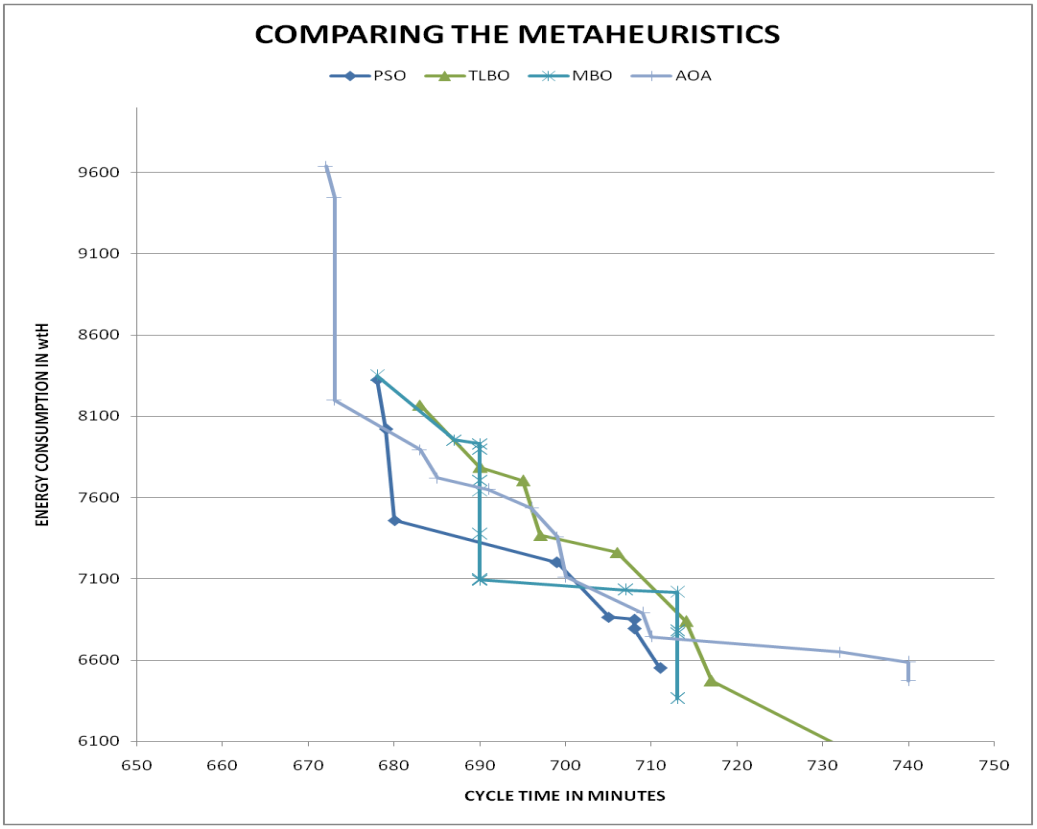
PROBLEM SIZE	Algo 1	Algo 2	Algo 3	Algo 4	Algo 5	Algo 6	Algo 7	Algo 8	OVERALL
Small	0	0	0	0	0	2	5	3	ALGO 7
Medium	0	0	0	1	2	1	4	6	ALGO 8
Large	0	0	0	0	3	3	6	6	ALGO 8

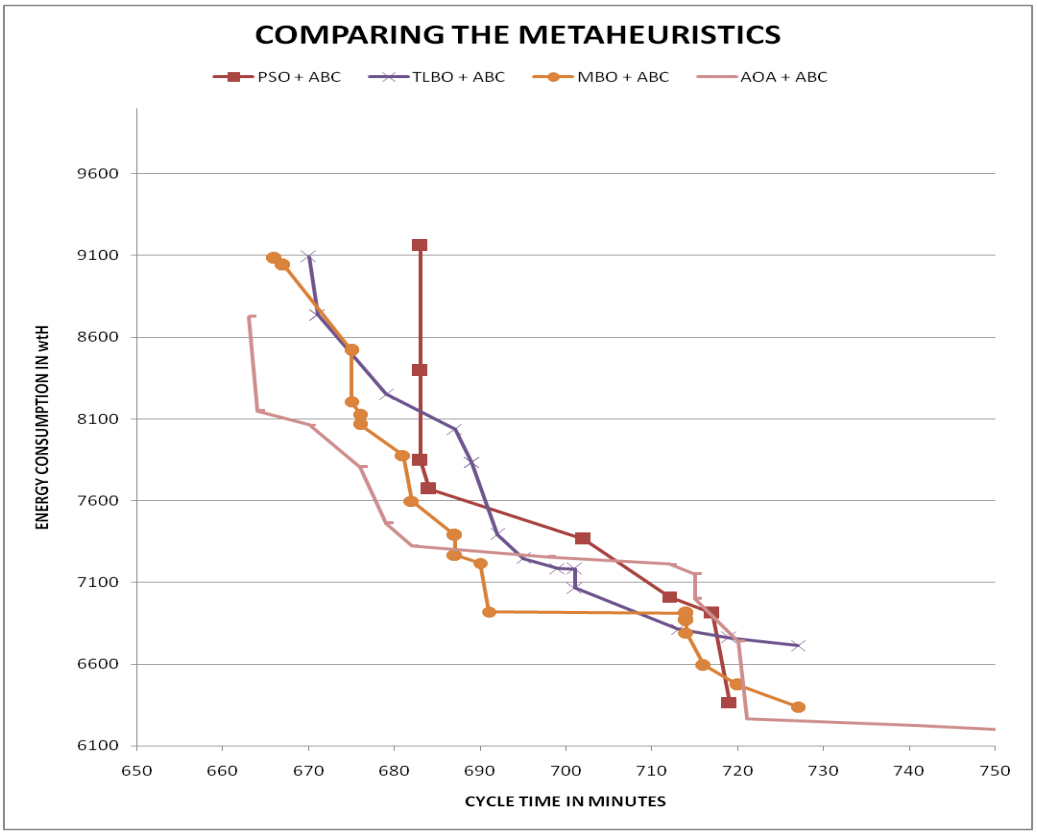
8.5 VISUALISATION OF RESULTS IN THE PARETO FRONT:




COMPARING THE METAHEURISTICS
PSO + ABC
TLBO + ABC
MBO + ABC
AOA + ABC
ENERGY CONSUMPTION IN wtH
CYCLE TIME IN MINUTES
1300
1200
1100
1000
900
800
700
600
950
1000
1050
1100
1150
1200
1250


COMPARING THE METAHEURISTICS
PSO
TLBO
MBO
AOA
ENERGY CONSUMPTION IN wtH
CYCLE TIME IN MINUTES
3250
3050
2850
2650
2450
2250
2050
1850
440
450
460
470
480
490
500
510
520
530
540


COMPARING THE METAHEURISTICS
PSO
TLBO
MBO
AOA
ENERGY CONSUMPTION IN wtH
9600
9100
8600
8100
7600
7100
6600
6100
650
660
670
680
690
700
710
720
730
740
750
CYCLE TIME IN MINUTES


COMPARING THE METAHEURISTICS
PSO + ABC
TLBO + ABC
MBO + ABC
AOA + ABC
ENERGY CONSUMPTION IN wtH
9600
9100
8600
8100
7600
7100
6600
6100
650
660
670
680
690
700
710
720
730
740
750
CYCLE TIME IN MINUTES

CHAPTER 9

WORK TO BE DONE

1. To apply the developed algorithms for a case-study in real-time industrial assembly data through a permitted industrial visit.

2. Modify the algorithms according to industry-specific norms and present the balanced assembly line results.

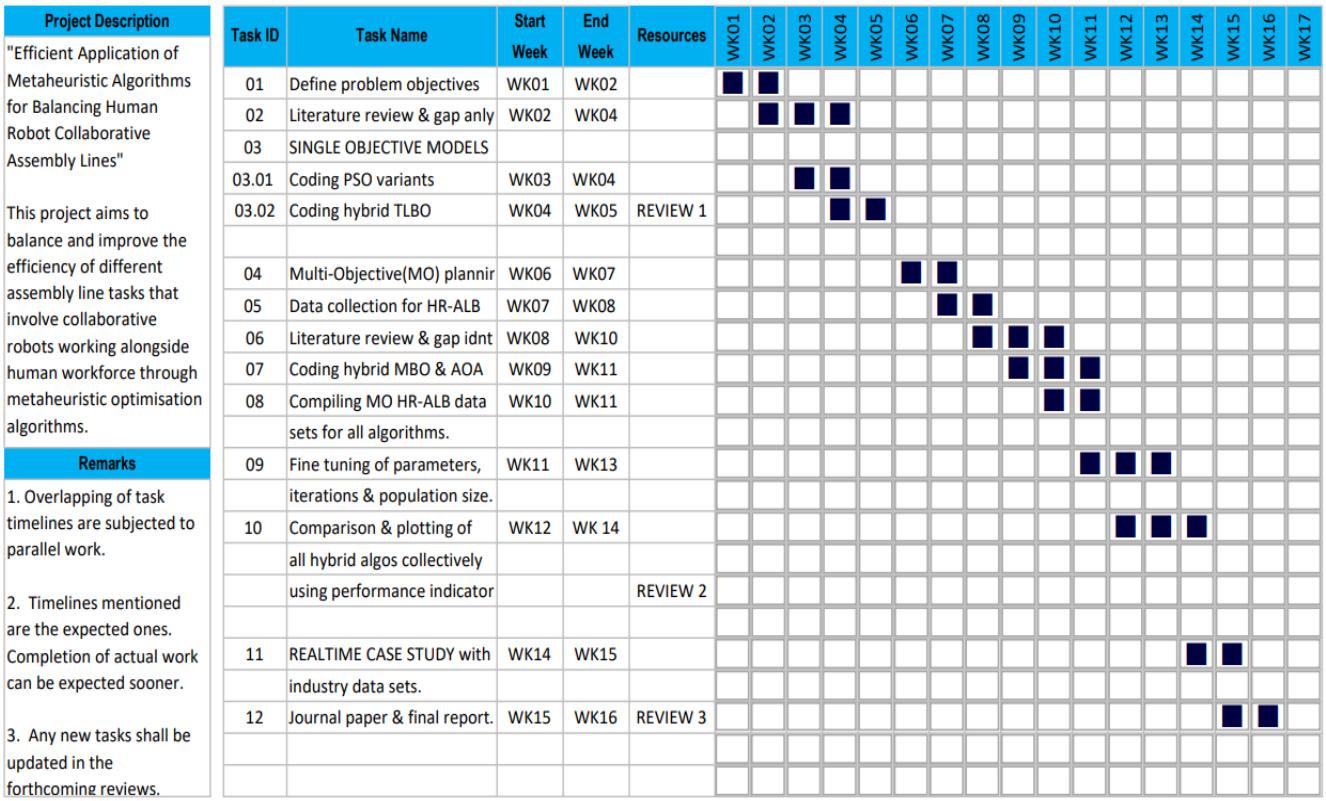
3. Completion of final report and journal for publication.

CHAPTER 10

MILESTONES AND GANTT CHART (WORK PLAN)

Project Description
"Efficient Application of Metaheuristic Algorithms for Balancing Human Robot Collaborative Assembly Lines" This project aims to balance and improve the efficiency of different assembly line tasks that involve collaborative robots working alongside human workforce through metaheuristic optimisation algorithms.
Remarks
1. Overlapping of task timelines are subjected to parallel work. 2. Timelines mentioned are the expected ones. Completion of actual work can be expected sooner. 3. Any new tasks shall be updated in the forthcoming reviews.

Task ID	Task Name	Start Week	End Week	Resources
01	Define problem objectives	WK01	WK02	
02	Literature review & gap anly	WK02	WK04	
03	SINGLE OBJECTIVE MODELS			
03.01	Coding PSO variants	WK03	WK04	
03.02	Coding hybrid TLBO	WK04	WK05	REVIEW 1
04	Multi-Objective(MO) plannir	WK06	WK07	
05	Data collection for HR-ALB	WK07	WK08	
06	Literature review & gap idnt	WK08	WK10	
07	Coding hybrid MBO & AOA	WK09	WK11	
08	Compiling MO HR-ALB data	WK10	WK11	
	sets for all algorithms.			
09	Fine tuning of parameters,	WK11	WK13	
	iterations & population size.			
10	Comparison & plotting of	WK12	WK 14	
	all hybrid algos collectively			
	using performance indicator			REVIEW 2
11	REALTIME CASE STUDY with	WK14	WK15	
	industry data sets.			
12	Journal paper & final report.	WK15	WK16	REVIEW 3



REFERENCES

[1] Weckenborg, C., Kieckhäfer, K., Müller, C. *et al.* Balancing of assembly lines with collaborative robots. *Bus Res* 13, 93–132 (2020). https://doi.org/10.1007/s40685-019-0101-y

[2] Tamás Koltai, Imre Dimény, Viola Gallina, Alexander Gaal, Chiara Sepe, An analysis of task assignment and cycle times when robots are added to human-operated assembly lines, using mathematical programming models, International Journal of Production Economics, Volume 242, 2021, 108292, ISSN 0925-5273, https://doi.org/10.1016/j.ijpe.2021.108292.

[3] Nicole Berx, Wilm Decré, Ido Morag, Peter Chemweno, Liliane Pintelon, Identification and classification of risk factors for human-robot collaboration from a system-wide perspective, Computers & Industrial Engineering, Volume 163, 2022, 107827, ISSN 0360-8352, https://doi.org/10.1016/j.cie.2021.107827.

[4] Tansel Dokeroglu, Ender Sevinc, Tayfun Kucukyilmaz, Ahmet Cosar, A survey on new generation metaheuristic algorithms, Computers & Industrial Engineering, Volume 137, 2019, 106040, ISSN 0360-8352, https://doi.org/10.1016/j.cie.2019.106040.

[5] Amir Nourmohammadi, Masood Fathi, Amos H.C. Ng, Balancing and scheduling assembly lines with human-robot collaboration tasks, Computers & Operations Research, Volume 140, 2022, 105674, ISSN 0305-0548, https://doi.org/10.1016/j.cor.2021.105674.

[6] Ana Correia Simões, Ana Pinto, Joana Santos, Sofia Pinheiro, David Romero, Designing human-robot collaboration (HRC) workspaces in industrial settings: A systematic literature review, Journal of Manufacturing Systems, Volume 62, 2022, Pages 28-43, ISSN 0278-6125, https://doi.org/10.1016/j.jmsy.2021.11.007.

[7] Li, Zixiang & Janardhanan, Mukund Nilakantan & S.G., Ponnambalam. (2021). Cost-oriented robotic assembly line balancing problem with setup times: multi-objective algorithms. Journal of Intelligent Manufacturing. 32. 10.1007/s10845-020-01598-7.

[8] Ya-jun Zhang, Ningjian Huang, Rober G. Radwin, Zheng Wang & Jingshan Li (2022) Flow time in a human-robot collaborative assembly process: Performance evaluation, system properties, and a case study, IISE Transactions, 54:3, 238-250, DOI: 10.1080/24725854.2021.1907489

[9] Zeynel Abidin Çil, Zixiang Li, Suleyman Mete, Eren Özceylan, Mathematical model and bee algorithms for mixed-model assembly line balancing problem with physical human–robot collaboration, Applied Soft Computing, Volume 93, 2020, 106394, ISSN 1568-4946, https://doi.org/10.1016/j.asoc.2020.106394.

[10] Li, Z., Janardhanan, M.N. & Tang, Q. Multi-objective migrating bird optimization algorithm for cost-oriented assembly line balancing problem with collaborative robots. *Neural Comput & Applic* 33, 8575–8596 (2021). https://doi.org/10.1007/s00521-020-05610-2

[11] Hashim, F.A., Hussain, K., Houssein, E.H. *et al.* Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems. *Appl Intell* 51, 1531–1551 (2021). https://doi.org/10.1007/s10489-020-01893-z