Swarm and Evolutionary Computation

Balancing Human-Robot Collaborative Assembly Lines using Metaheuristic Algorithms with a Case-study in an electronic assembly unit

--Manuscript Draft--

Manuscript Number:	SWEVO-D-22-00303
Article Type:	Full Length Article
Keywords:	Assembly Line balancing; Human-robot collaboration; Collaborative robots; multi-objective optimization; hybridization; metaheuristics
Corresponding Author:	Ponnambalam S.G. Vellore Institute of Technology (VIT) Vellore, Tamil Nadu INDIA
First Author:	Ranganathan S.V.
Order of Authors:	Ranganathan S.V.
	Sridharan A.P.
	Ponnambalam S.G.
	Mukund Nilakantan Janardhanan
Abstract:	In this paper, eight metaheuristic algorithms are proposed to balance the human-robot collaborative (HRC) assembly line with the bi-objectives of minimizing cycle time and energy consumption. To The literature on HRC assembly lines is very minimal and the current research will be valuable for the HRC research community. Four metaheuristic algorithms namely, Teaching-learning-based optimization (TLBO), Particle swarm optimization (PSO), Migrating-birds optimization (MBO) and Archimedes optimization algorithm (AOA) are considered and each of these four algorithms are hybridized with the scout phase of artificial bee colony (ABC) algorithm yielding total of eight algorithms. Novel parameter fine-tuning methods are proposed to reach better solutions and to minimize the computational efforts. The Pareto optimal solutions of the metaheuristic algorithms compared using three performance indicators namely, non-dominated solution count, global error ration and convergence metric. To date, no benchmark datasets are available for HRC assembly line problems. A novel methodology is proposed to generate datasets and using this methodology thirty-two HRC datasets of varying sizes and workstations are generated and used to evaluate the algorithms, which could be useful for the HRC research community. It is observed among the eight algorithms, AOA algorithm performs better than other algorithms with minimal computational effort. Statistical analysis is conducted using Friedman's test and Scheffé multiple comparison test as the post-hoc test. The results of these testes confirms the better performance of AOA algorithm. A case study is carried out in an electronic assembly unit to validate the research done and the results are presented.
Suggested Reviewers:	Qiuhua Tang, Ph.D. Professor, Wuhan University of Science and Technology tangqiuhua@wust.edu.cn Professor Qiuhua Tang's research area includes application of metaheuristic algorithm to balance assembly lines. He regularly published articles in this area in reputed journals such as Swarm and Evolutionary Computation, Journal of Intelligent Manufacturing and so on.
	Loo Chu Kiong, Ph.D professor, University of Malaya Library: Universiti Malaya ckloo.um@um.edu.my Professor Loo's research area includes swarmrobotics and evolutionary algorithms. He is extensively publishing articles in these areas in reputed journals. He has published more than 100 articles in reputed journals.
	Mohd Fadzil Faisae B Ab. Rashid, Ph.D. Associate Professor, University Malaysia Pahang: Universiti Malaysia Pahang ffaisae@ump.edu.my Dr. Mohd Fadzil Faisae's research areas include assembly line balancing and

	metaheurtic algorithms. He has published more than 50 articles in these area in reputed journals.
Opposed Reviewers:	

Balancing Human Robot Collaborative Assembly Lines using Metaheuristic Algorithms with a Case-study in an electronic assembly unit

S.V.Ranganathan[1], A.P. Sridharan[1], S.G. Ponnambalam[1*], Mukund Nilakantan Janardhanan[2]

[1]School of Mechanical Engineering, Vellore Institute of Technology, Vellore-632014, India, email: rangu230801@gmail.com, sridharsir30@gmail.com, ponnambalam.g@vit.ac.in

[2]School of Engineering, University of Leicester, University Road, Leicester, LE1 7RH, UK, email: mukund.janardhanan@leicester.ac.uk

*corresponding author (email: ponnambalam.g@vit.ac.in)

Balancing Human Robot Collaborative Assembly Lines using Metaheuristic Algorithms with a Case-study in an electronic assembly unit

Abstract

In this paper, eight metaheuristic algorithms are proposed to balance the human-robot collaborative (HRC) assembly line with the bi-objectives of minimizing cycle time and energy consumption. To The literature on HRC assembly lines is very minimal and the current research will be valuable for the HRC research community. Four metaheuristic algorithms namely, Teaching-learning-based optimization (TLBO), Particle swarm optimization (PSO), Migrating-birds optimization (MBO) and Archimedes optimization algorithm (AOA) are considered and each of these four algorithms are hybridized with the scout phase of artificial bee colony (ABC) algorithm yielding total of eight algorithms. Novel parameter fine-tuning methods are proposed to reach better solutions and to minimize the computational efforts. The Pareto optimal solutions of the metaheuristic algorithms compared using three performance indicators namely, non-dominated solution count, global error ration and convergence metric. To date, no benchmark datasets are available for HRC assembly line problems. A novel methodology is proposed to generate datasets and using this methodology thirty-two HRC datasets of varying sizes and workstations are generated and used to evaluate the algorithms, which could be useful for the HRC research community. It is observed among the eight algorithms, AOA algorithm performs better than other algorithms with minimal computational effort. Statistical analysis is conducted using Friedman's test and Scheffé multiple comparison test as the post-hoc test. The results of these testes confirms the better performance of AOA algorithm. A case study is carried out in an electronic assembly unit to validate the research done and the results are presented.

Keywords: Assembly line balancing; Human–robot collaboration; Collaborative robots; multi-objective optimization; hybridization; metaheuristics

1. Introduction

In any industry, the optimization of assembly lines mostly starts with identifying and balancing several sub-assembly processes to break down the study complexity or in automating a set of tasks under economic considerations. In considering the current approaches to solving ALB problems, the first category deals with maximization of throughput in the assembly line with a given number of resources (Dalle Mura and Dini 2022). Whereas the second category is contrary to the first and deals with minimization of cost of resources for a fixed throughput number (Boysen, Schulze, and Scholl 2021). The studies conducted by Hamzadayi (Hamzadayi 2018) and Qiuhua (2017) demonstrate the necessity of balancing two-sided assembly lines, where large assembly models require work-handling in every direction and the orientation setup times are very large, which may prove infeasible. This shows that the best layout is mostly dependent on the type and count of products being assembled.

1.1 ROBOTIC ASSEMBLY LINE BALANCING (RALB)

With the introduction of artificial intelligence (AI) and commercial automation technology, the simple justification to replace them with their manual counterparts is due to their accuracy, speed, flexibility, and long-term cost benefits involved (Malik and Bilberg 2019). The robotic assembly lines, therefore, lead to high volume and stable production levels. From the knowledge gained during the industrial case study implementation done later in this paper, the initial cost of automation deployment is on terms such that the cost is recovered within 20% of the total project timeline. Hence, a vital part of RALB also lies in automating existing manual processes by considering ROI and task feasibility. The general ALB problem has various objectives such as minimizing cycle time, cost, increasing safety, ergonomics and so on, with constraints such as task precedence relations, number of workstations, resources etc. As industrial robots and gantries are accurate in performing tasks, they are assumed to have deterministic operation times in solving RALB problems (Chutima 2022). The RALB type-I aims to reduce the workstation count by assigning best suited robots and tasks to workstations; RALB type-II aims at minimizing the cycle time with predefined workstation counts; RALB type-E aims at reducing cycle time and workstation count together, thus maximizing the assembly line efficiency; RALB type-F aims at finding feasible solutions for a predefined set of workstations and cycle times; RALB type-COST aims at monetary and economic aspects; RALB type-O involves other categories not listed above (Koltai et al. 2021; Weckenborg et al. 2020).

1.2 HUMAN-ROBOT COLLABORATIVE ASSEMBLY LINE BALANCING

To combine the flexible decision making of manual labor with the accuracy of robots, the study of line balancing in several workspaces considers both humans and robots working hand in hand. This method of allocation can induce a great extent of intelligent manufacturing capability and flexibility to assembly systems (Weckenborg et al. 2020; Simões et al. 2022). Berx et al. (Berx et

al. 2022) suggests careful assessment of every individual task as a primary step before deciding on its implementation ability by either humans, robots or collaborative robots (cobots). By also keeping in mind the cost of automation and margin of lead time reduction, the ranking of risks associated with every task and resource combinations can be documented, to further decide on alternative allocation criteria and help strike a perfect balance. The known possibilities of human-robot interactions in a workstation can be stated by considering the following scenarios: (1) If the human worker is present in the opposite lane to the robot, where both work on the same task; (2) If the human and robot are in a single workstation, but work in shifts and don't coexist; (3) If both human and robot are present in a single workstation but do not work simultaneously; (4) If both the resources work simultaneously/does parallel work (Gualtieri, Rauch, and Vidoni 2021). Setting aside the first two scenarios, scenario (4) is largely neglected by several prominent studies in this field (Nourmohammadi, Fathi, and Ng 2022; Li, Janardhanan, and Ponnambalam 2021). This is due to safety considerations that parallel tasking of both humans and robots may overlap and cause disruption. However, it is worthwhile to point out that the extent of error possibility largely depends on the sensitivity of tasks in the workspace. Additionally, parallel working lessens the interaction between automation and humans, thus increasing the error occurrence possibility due to reduced manual maintenance (Dalle Mura and Dini 2019).

2. Literature review

Several different scenarios of prerequisite input conditions/constraints are considered for every ALB research problem, some of which include precedence constraints, complexity ranking and classification of tasks, operational and setup times, assumptions of resource availability or other constraints, all in the interests of the type of assembly line under study (Simões et al. 2022). Research emphasis is still downside on data retrieval models that serve as a foundation to formulating line balancing problems. Machine learning predictive modean be useful in deciding the “at least near” precedence relation sequence, to aid as input data for algorithmic ALB (Boysen, Schulze, and Scholl 2021). As the area of research in human-robot assembly lines is quite new, only a few papers have been published since 2019. Only a handful of studies addressed the environmental issues (carbon footprint) by the robots (Hamzadayi 2018). The concept of 4 M's are kept in mind while considering the several resources of an assembly line, Man, Machine, Material & Method (Borba, Ritt, and Miralles 2018). Several methods are used to arrive at a solution set, and some of them are exact optimization techniques, using heuristics, and meta-heuristics (Chutima 2022). As the automation in the manufacturing industry is increasing significantly, robots are deployed to carry out several tasks. However, not all tasks can be automated due to the lack of flexibility with available robotic technologies (Stecke and Mokhtarzadeh 2022). Humans are not only flexible and adaptable according to market demands, but also have decision making skills with creativity. For the assignment of the cobots to different stations with balanced distribution of workload to both human workers and the robotic partners, MILP methods are widely used although it sets a major drawback in problem size limitations (Weckenborg et al. 2020).

The major concern with automation is their purchasing costs (Li, Janardhanan, and Tang 2021), which indirectly correlates with the utilization efficiency and power consumption. Most industries ensure that the procurement costs are worth the investments on remunerative contract projects they deal with and also ensure the terms of ROI in the near term. As most portion of the literature only considers non-parallel working of both human and robot in a single workstation, the study conducted by Zhang et al. (2022) facilitates a stochastic line balancing model where a portion of the tasks are allowed parallel work handling between the two resources. This can prove to be the bare minimum requirements that can be considered in balancing studies that might want the parallel working of humans and robots on the same component (Vieira et al. 2021). The recent studies in HRC ALB areas focus on three allocation possibilities: (1) only workers are present in workstations; (2) either workers or robots are present in the same workstation; (3) both workers and robots are present in a workstation. For multi-objective optimization approaches for the same, either the idealistic objective value is fixed, and the required resources are determined or alternatively, the optimal objective value is determined for the available amount of resources (Koltai et al. 2021).

During task allocation between robots and humans, skill requirements for all tasks are assessed to assign robots with tasks pertaining to low skill requirements, high complexity, precision, and repetitiveness; and assign human workers with ergonomic tasks of little complexity and high cognitive adaptability skills to adapt to an error-free operative environment and suppress the risk of automation (Berx et al. 2022; Weckenborg and Spengler 2019). Metaheuristics largely aid in quick computations of line balancing problems that consider several constraints and combinatorial possibilities in simultaneous task and resource allocation along the assembly lines (Dokeroglu et al. 2019). Population based metaheuristic algorithms prove superior performances (Taheri, RahimiZadeh, Rao 2021, and Li et al. 2019) and also a long chain of multiple line balancing implementations that have been carried out over the years (Nourmohammadi, Fathi, and Ng 2022; Li, Janardhanan, and Ponnambalam 2021; Ab Rashid, Hutabarat, and Tiwari 2018; Tang et al. 2017). Similar to line balancing, studies have demonstrated several other optimization applications in various fields (Zhang et al. 2020; Kashani et al. 2021; Parsopoulos and Vrahatis) which mainly used the PSO and TLBO variants. These studies introduce variants and modifications to the standard algorithms to better suit the specific problem environment or to improve the existing mechanisms (Taheri, RahimiZadeh, and Rao 2021; Sun and Gao 2019). The basic intention behind introduction of modifications by combining various other nature inspired meta-heuristic approaches are to: (1) Enhance exploration; (2) Converge to optima in an appropriate time to improve efficiency; (3) Avoid premature convergence towards local optima; (4) increase the overall logical suitability of the approach, based on the optimization problems. To handle multi-objective optimization problems, the NSGA-II technique is the most widely used, with applications in novel HRC studies. For this reason, this study also utilizes a reduced computational form of non-dominated sorting (NDS) as given in the NSGA-II formulations (Reddy and Kumar 2006; Deb et al. 2002)**.** Tang et al. (2022) proposed a multi-objective multifactorial evolutionary

algorithm to minimize cycle times and operation alterations in assembly lines considering regular production and preventive maintenance scenarios. Specifically, the native evolutionary operators are reformulated to encourage the inter-scenario sharing of effective knowledge fragments among allocation plans to accelerate the simultaneous optimization of all the tasks.

By far, it is clear from literature that studies on collaborative line balancing lack considerations of line energy consumption, which is a key factor in determining manufacturing costs and efficiency. The few existing studies on HRC focus very little on the complex discretization methods that are often raveled upon while decoding the problem environment through metaheuristic algorithms.

Metaheuristics provide excellent methodologies in solving optimization problems that involve a multitude of datasets. It remains an active area of research, where new updating mechanisms are being proposed based on natural observations. It varies from continuous to discrete computations and provides the flexibility of including constraints within its mechanism. Because of their intricate nature, solving these challenges is not an easy process. Metaheuristic algorithms are meant to find approximate solutions in realistic execution times for NP-hard optimization problems and give a practical and elegant answer to many such situations (Dokeroglu et al. 2019). In general, metaheuristics draws upon its motivation from nature-based interaction techniques such as the collective swarm behavior among animals or the daily interactions towards improvement or evolutionary techniques per say. The metaheuristic algorithms that have been considered for this study include PSO, TLBO, MBO and AOA as their embedded convergence mechanisms provide great flexibility in modifying or creating variants, to suit the specific problem that is being considered. An induction of a new phase/strategy further improves the performance of the standard algorithm and covers a gap of possibilities between existing metaheuristic algorithms.

3. Problem description

The problem considered in this paper is based on the literature and realistic considerations of line ergonomics, safety and intellectual diversity which is supported by a case-study done in the electronics industry. As like any other HRC line balancing problem, this problem considers three possibilities of assembly line work setting: tasks operated by a human alone, tasks operated by concerned robots alone, and tasks operated by both human worker and robot in collaboration (termed as "cobot" in our problem). As in any industry, all HRC resources (humans/individual robots and cobots) considered for this problem are strictly constrained based on their abilities/possibilities in carrying out each task. As shown in **table 1**, only the task operations assigned with '1' can be carried out by the respective resource.

After careful considerations from literature and the case-study conducted, and to not neglect any possibility and choice for readers, two scenarios in the HRC resource availability have been

considered for this problem: 1) The primary consideration is such that, there is an infinite count of robots of required types and human workers available for allocation among the given number of workstations. 2) There is a known count of robots of each type and an infinite count of human workers.

Although the first scenario lacks economic considerations to an extent, it is followed by a resource allocation constraint for the given number of workstations. This constraint is introduced to develop an optimized allocation sequence for an evenly spread semi-automated assembly line. The constraint is such that no two consecutive workstations shall have the same resource configurations. **Figure 1** shows an assembly line illustration where the workstation constraint is considered. Each of the three possibilities depict a different possible combination of resource allocation, in all of which no two consecutive stations have the same type of resource.

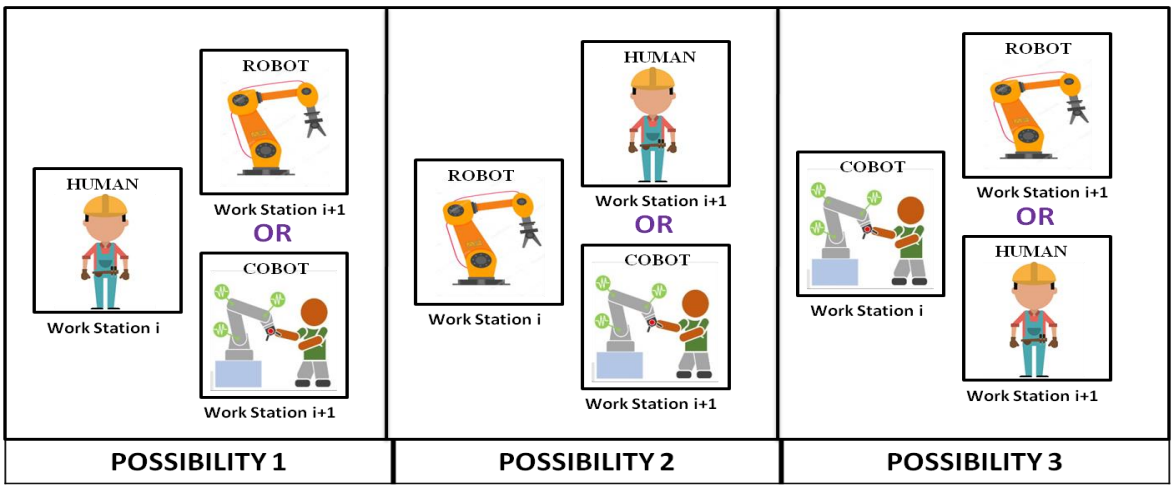

Figure 1. Illustration of proposed workstation constraint

Based on the problem considered by Li et al. (2021), the process alternatives sequence shall be used to execute the resource allocation constraint successfully. Regarding several task executions within a single workstation, it is to be noted that this problem considers only one type of resource allocation per workstation (either one human, one robot or one cobot) and neglects the parallel working of cobot elements, which would otherwise compromise robustness in the light of anomalies. The problem could, however, accommodate parallel working of cobot elements under management considerations. For this problem, data pertaining to prior known operational times, setup times and so forth are randomly generated based on upper and lower bound values, as to account for task time fluctuations in the assembly lines and also to reduce fatigue in real-world HRC data generation.

The assumptions considered for this study are,

- This study considers only a straight-line assembly line (StAL) where a single model is assembled.
- Number of assembly workstations is known prior.
- Skill sets of all human workers are the same.
- Task possibility data and individual operation times of all types of robots and human workers in terms of upper and lower bounds are known prior.
- Sequence dependent times and setup times of all resources are known prior.
- Precedence relations of all tasks are known.
- Energy ratings of robots are known (can be considered for cost analysis along with labor charges of human workers)
- Robot maintenance is assumed to be done during non-production hours. In the event of any robot breakdown/down-time, the workstation(s) equipped with the failed robot(s) are simply removed to generate a new temporary emergency task allocation sequence with the humans.

4. Data representation/generation

In line with the basic data requirements in solving assembly line problems in HRC environment, the inputs required are: 1) Total number of tasks; 2) Total workstation count; 3) Number of different robot models; 4) Task operation times for each robot models, individual human worker, and cobot; 5) Setup times and sequence dependent times for each robot model; 6) Precedence relation of all tasks; 7) Energy rating of every robot models.

Due to the lack of standard datasets for HRC, a module [C++ code] is developed for the generation of data according to the methodology below, which also conforms to the norms of the HRC problem environment. To generate the above stated data for this problem, an unique methodology is proposed as follows:

- Number of different cobot models = Number of different robot models.
- A cobot model [type i] is equivalent to a robot [type i] working with a human worker.
- Only one type of human worker is assumed (all workers are equally skilled).
- The lower bound $(\boldsymbol{lb_i})$ and upper bound $(\boldsymbol{ub_i})$ of operation time for each robot model is inputted by the user and is used for generating all the remaining data (operation time for cobot & human, setup and sequence dependent time of each type of robot).
- With the consideration of human worker flexibility, the setup time and sequence dependent time for human workers are neglected.

The bounds are generated using four parameters, a1, a2, a3 & a4 which values belong in the range [0,1]. For generating the range for human workers, the average lower bound and upper bound of

all the types of robots is used. The relations used to find the boundary values are given below where, $\boldsymbol{lb_a}$ refers to the average of lower bounds of all types of robots and $\boldsymbol{ub_a}$ refers to the average of upper bounds of all types of robots.

- Lower limit of task time for robot [type i] - $(\boldsymbol{lb_i})$
- Upper limit of task time for robot [type i] - $(\boldsymbol{ub_i})$
- Lower limit of setup time for robot [type i] - $(\boldsymbol{a1} * \boldsymbol{lb_i})$
- Upper limit of setup time for robot [type i] - $(\boldsymbol{a1} * \boldsymbol{ub_i})$
- Lower bound of sequence dependent time for robot [type i] - $(\boldsymbol{a2} * \boldsymbol{lb_i})$
- Upper bound of sequence dependent time for robot [type i] - $(\boldsymbol{a2} * \boldsymbol{ub_i})$
- Lower limit of task time for cobot [type i] - $\boldsymbol{lb_i} - (\boldsymbol{a3} * (\boldsymbol{ub_i} - \boldsymbol{lb_i}))$
- Upper limit of task time for cobot [type i] - $\boldsymbol{ub_i} - (\boldsymbol{a3} * (\boldsymbol{ub_i} - \boldsymbol{lb_i}))$
- Lower limit of task time of human - $\boldsymbol{lb_a} + (\boldsymbol{a4} * (\boldsymbol{ub_a} - \boldsymbol{lb_a}))$
- Upper limit of task time of human - $\boldsymbol{ub_a} + (\boldsymbol{a4} * (\boldsymbol{ub_a} - \boldsymbol{lb_a}))$

To avoid the laborious manual data entry, an automated procedure has been developed using the above relations to generate the datasets. However, the manual entry of data can be done.

A set of tabulations given in this section provide an instance for data generation of a 5 task problem with 3 different robot types, each of single quantity. The precedence relations of tasks and sequence dependent times are illustrated in **table 2**. The parameters a1, a2, a3 & a4 are instantiated in the beginning and are taken as 0.25, 0.1, 0.5 & 0.3 respectively, in order to perform the necessary calculations. Figure 2 illustrates the calculation of bounds for the 3 robots example problem. The same relations are used to generate the operation times of each robot, cobot pairs and the human worker, as

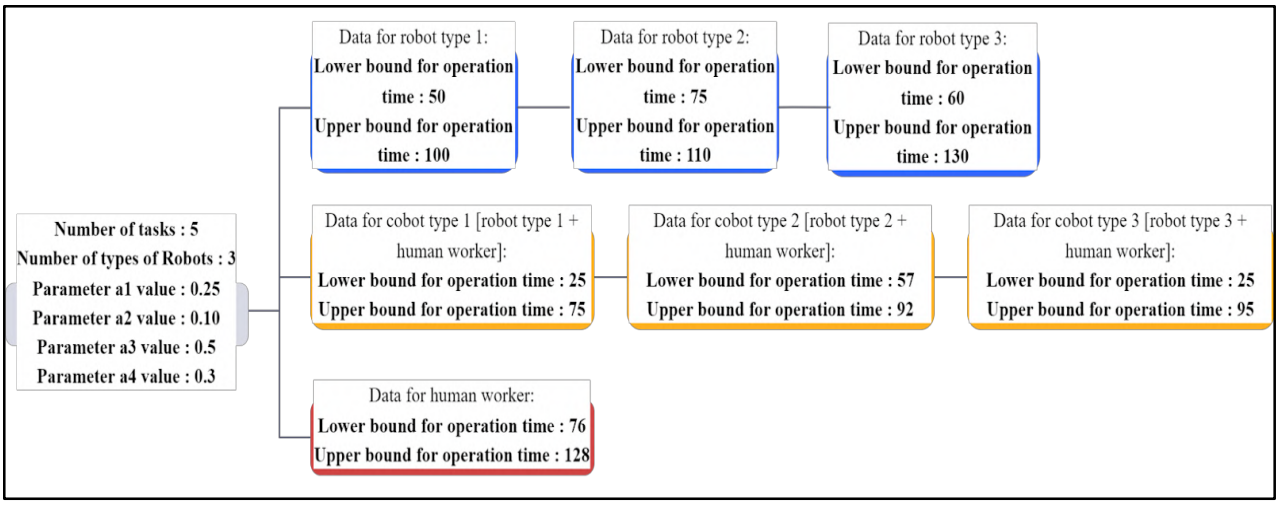

Figure 2. Illustration of example input values for data generation

shown in **table 1**. The blank spaces in the table signify that the task implementation by a particular resource for a given task is not possible. The setup times and sequence dependent setup times for every resource type are displayed with the help of the range bound equations considered. Generally, the energy rating for robots with a payload of 7kg (https://robotsdoneright.com/FANUC/LR-Mate-Series/FANUC-LR-Mate-200id.html) is assumed in the range of [0.2,0.4] kW.

5. Population Initialization and function evaluation

Each solution in the population consists of two strings, one represents the task sequence and the other represents the process alternative linked to the task sequence. Each task sequence vector represents the combinations of task numbers ordered randomly and which strictly follow precedence relations. Topological sorting procedure is used, which uses the adjacency list to generate the task sequences (Ab Rashid, Hutabarat, and Tiwari 2018). Process alternative vector contains the resource allocation identity numbers 1 or 2 or 3 [1 means human, 2 means Robot of type_i and 3 means Cobot of type_i] for each workstation. For this study, the process-alternative vectors follow the workstation constraint mentioned in the problem description by not having the same identity number in any two consecutive workstations. The topological sorting procedure is given in *algorithm-1*. Process alternative string is generated randomly in the range of [1-3] for each workstation.

ALGORITHM 1:Pseudocode for topological sorting to generate Task Sequences

n = 0; *number of tasks in generated sequence (initially no tasks are in the vector)*
N; *total number of tasks that are to be allocated*
Create an empty vector - ***task sequence vector;*** *holds the generated task sequence*
Create an ***available set;*** *consists of tasks that has no precedence*
While *(n <= N)****:***
    Pick a random task from the available set
    Insert this task into the task sequence vector
    Remove this task from the available set
    Insert the tasks that had this removed task as its precedence
    Increment n by 1
End While

An example in **figure 3** portrays both task sequence and process alternative sequence, considering the workstation constraint mentioned earlier. For the calculation of cycle time after the allocation of a particular task sequence and process alternative among workstations, the consecutive heuristic method proposed by Levitin et al. (2006) is used.

Table 1. Task operation times of HRC resources (example data)

RESOURCES	FUNCTION	*Task 1*	*Task 2*	*Task 3*	*Task 4*	*Task 5*
Robot 1	OPERATION TIME	56	68	82	94	---
	TASK CONSTRAINT	1	1	1	1	0
	SETUP TIME	24	23	13	19	---
Robot 2	OPERATION TIME	76	---	93	---	90
	TASK CONSTRAINT	1	0	1	0	1
	SETUP TIME	18	---	---	21	21
Robot 3	OPERATION TIME	102	111	78	---	---
	TASK CONSTRAINT	1	1	1	0	0
	SETUP TIME	22	24	29	---	---
Cobot 1 (robot 1 + human)	OPERATION TIME	48	70	71	48	74
	TASK CONSTRAINT	1	1	1	1	1
	SETUP TIME	---	---	---	---	---
Cobot 2 (robot 2 + human)	OPERATION TIME	67	76	85	80	60
	TASK CONSTRAINT	1	1	1	1	1
	SETUP TIME	---	---	---	---	---
Cobot 3 (robot 3 + human)	OPERATION TIME	74	62	68	26	29
	TASK CONSTRAINT	1	1	1	1	1
	SETUP TIME	---	---	---	---	---
Human	OPERATION TIME	---	117	---	96	89
	TASK CONSTRAINT	0	1	0	1	1
	SETUP TIME	---	---	---	---	---

Table 2. Sequence dependent times & precedence matrix of HRC resources (example data)

Tasks		*Task 1*	*Task 2*	*Task 3*	*Task 4*	*Task 5*
Task 1	SEQUENCE DEPENDENT TIME FOR ROBOT 'i'	7	7	8	5	8
	PRECEDENCE	0	1	1	0	0
Task 2	SEQUENCE DEPENDENT TIME FOR ROBOT 'i'	8	8	7	6	7
	PRECEDENCE	0	0	0	1	0
Task 3	SEQUENCE DEPENDENT TIME FOR ROBOT 'i'	6	10	8	6	9
	PRECEDENCE	0	0	0	0	1
Task 4	SEQUENCE DEPENDENT TIME FOR ROBOT 'i'	5	10	5	10	10
	PRECEDENCE	0	0	0	0	1
Task 5	SEQUENCE DEPENDENT TIME FOR ROBOT 'i'	5	7	10	6	5
	PRECEDENCE	0	0	0	0	0

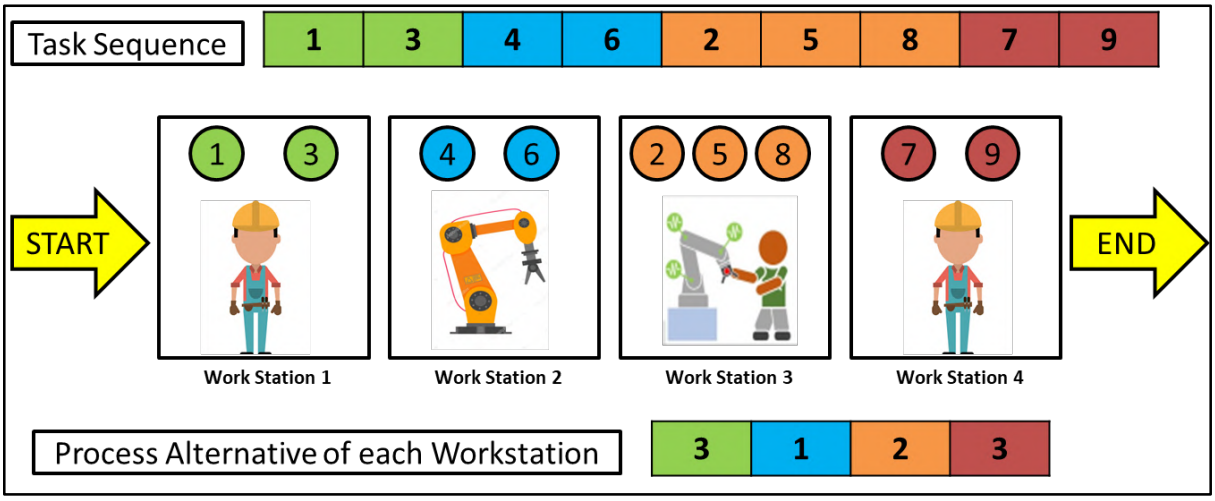

Figure 3. An example task sequence and linked process alternative

Sequence dependent setup times are considered separately in this paper. Figure 4 explains the calculation of cycle time with sequence dependent setup times.

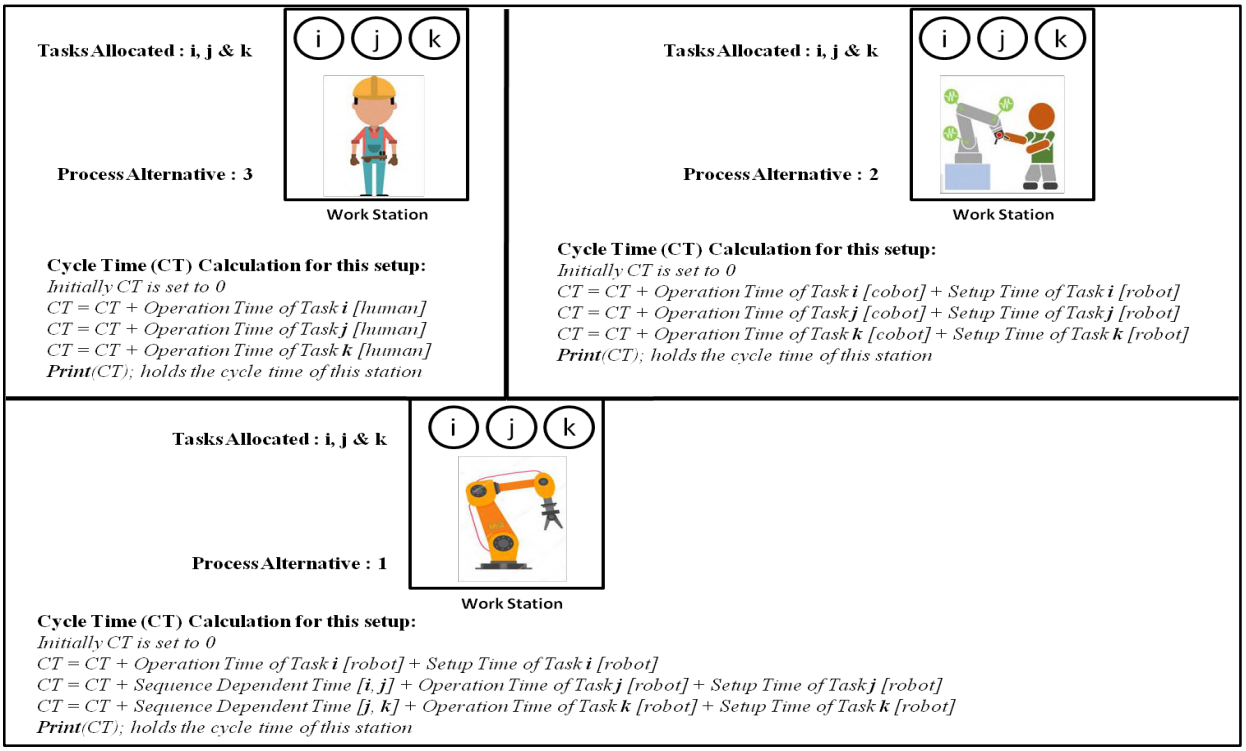

Figure 4. Cycle time calculation with sequence dependent setup times

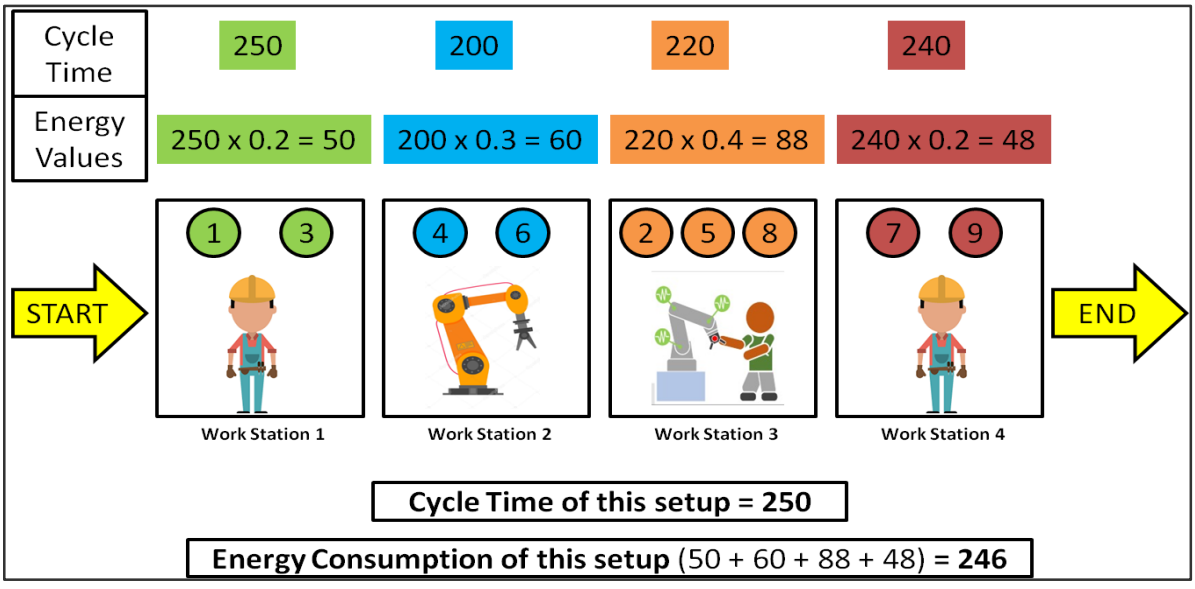

Figure 5. Cycle time & energy consumption calculations for the example (figure 3)

Using the consecutive heuristic method as mentioned in (Levitin et al, 2006)), cycle time for each workstation is calculated and the final CT (obtained after running the algorithm) is considered as

the cycle time of the given task sequence and process alternative vector. Parallelly, the energy consumption is calculated as part of the secondary objective by multiplying the energy rating of the process alternative with the cycle time of each workstation and is summed up, as shown in **figure 5.**

6. Proposed multi-objective algorithms

The four algorithms, namely PSO, TLBO, MBO and AOA, are considered in this paper. AOA is the most recently introduced algorithm of all, that hasn't been used in discrete environments and assembly line balancing areas. The attractiveness of the AOA algorithm is that no parameter fine tuning is required. All these algorithms extend their simplicity to be discretized and hybridized with scout-phase of artificial bee colony algorithm, thus gaining advantages in reaching better solutions. Thus, four algorithms and four hybrid algorithms are evaluated in this study.

To handle multiple objectives, the elite and fast non-dominated sorting (NDS) approach, which is known as NSGA-II proposed by Deb et al. (2002) is employed. As compared to a single-objective approach where there exists only a single feasible solution, a multi-objective approach can have conflicting situations. An external archive (EA) or a Pareto-optimal set holds the feasible solutions.

Generally, researchers use NSGA-II that generates the pareto front solutions after NDS to handle multi-objective problems (Li, Janardhanan, and Tang 2021; Ab Rashid, Hutabarat, and Tiwari 2018), which increases the computational effort. Based on the analysis conducted, it is found that only the front-1 solutions are sufficient to find the best solution for the problem. As all multi-objective studies use the concept of crowding distance, this study on the contrary picks only the best solution randomly from the corner solutions of the pareto front-1. This assumption holds true as the corner solutions hold high weightage criteria in the crowding distance method and significantly affect the computation and convergence of each algorithm.

6.1 DISCRETE PARTICLE SWARM OPTIMIZATION

Particle swarm optimization algorithm (PSO) is a population-based meta-heuristic algorithm perceived from swarm intelligence techniques to collectively explore the best habitat amongst their population, considering several objectives and cautions. Each population member plans their consequent move from their respective individual finest discovery as well as the collective best achievement. There are two stoichiometric coefficient-based equations, namely velocity update and position update (Kennedy and Eberhart, 1995). Both the velocity and positional update equations define the relation between the current and successive position and velocity of each particle, respectively. The positional vector equation is given by:

$$\boldsymbol{X_i^{t+1} = X_i^t + V_i^{t+1}} \qquad (\mathbf{1})$$

Where, $\boldsymbol{X_i^{t+1}}$ is the position of any population in (t+1)th iteration, which is the immediate position after $\boldsymbol{X_i^t}$ (position of the member in (t)th iteration). The $\boldsymbol{V_i^{t+1}}$ denotes the velocity of the successive iteration, wherein the velocity update equation is:

$$\boldsymbol{V_i^{t+1} = (c1 * V_i^t) + \left(c2 * \left(P_{best,i} - X_i^t\right)\right) + \left(c3 * (G_{best} - X_i^t)\right)} \quad (2)$$

The decoded task sequences and process alternative sequences are calculated of their objectives and are then plotted to undergo NDS, in order to find the front 1. In every consecutive iteration, the best solution of each individual particle (called $\boldsymbol{P_{best,i}}$) is updated as the position of each particle gets updated as shown in **equation (1).** These positional updates are driven by the velocity update equation as shown in **equation (2),** which contains $\boldsymbol{G_{best}}$ & $\boldsymbol{P_{best,i}}$ as parameters. The $\boldsymbol{G_{best}}$ is the best solution achieved by the entire swarm of particles at each iteration and is determined by randomly picking one corner solution of the front 1 found using NDS as explained earlier.

6.1.1 OPERATORS USED IN PSO

From the existing discretization approaches that have been used by Rashid et al. (Ab Rashid, Hutabarat, and Tiwari 2018) for the various PSO operators, this study uses the basic version of the same throughout this paper. The pseudocodes of the procedures are given below. The pseudocodes for the operators used in PSO are discussed below.

PSEUDOCODE FOR ADDITION OPERATOR-1 (Task Sequence + Velocity Vector)

V; *velocity vector*
X; *task sequence vector*
Y; *vector to hold the answer after carrying out the operation*
For *i in range(1, Number of tasks):*
    If*(V[i] is X[i] or V[i] is 0):*
        assign X[i] to Y[i]
    Else if*(V[i] != 0):*
        assign V[i] to Y[i]
    Else:
        assign 0 to Y[i]
    End if
End for

PSEUDOCODE FOR ADDITION OPERATOR-2 (Velocity Vector 1 + Velocity Vector 2)

V1; *velocity sequence 1*
V2; *velocity sequence 2*
Y; *vector to hold the answer after carrying out the operation*
For *i in range(1, Number of tasks):*
    If*(V1[i] is not 0 and V2[i] is not 0):*
        assign V2[i] to Y[i]
    Else:
        assign V1[i] to Y[i]
    End if
End for

PSEUDOCODE FOR SUBTRACTION OPERATOR-1 (Task Sequence 1 - Task Sequence 2)

X1; *task sequence 1*
X2; *task sequence 2*
Y; *vector to hold the answer after carrying out the operation*
For *i in range(1, Number of tasks):*
    If*(X1[i] is X2[i]):*
        assign X1[i] to Y[i]
    Else:
        assign X2[i] to Y[i]
    End if
End for

PSEUDOCODE FOR MULTIPLICATION OPERATOR (Co-efficient * Sequence Vector)

X; *task sequence 1*
C; *co-efficient (c1 / c2 / c3)*
Y; *vector to hold the answer after carrying out the operation*
For *i in range(1, Number of tasks):*
    R; *generate random number between (0 to 1)*
    If*(R < C):*
        assign X1[i] to Y[i]
    Else:
        assign 0 to Y[i]
    End if
End for

6.2 DISCRETE TEACHING LEARNING BASED OPTIMIZATION

The Teaching Learning Based Optimization (TLBO) as proposed by Rao et. al (2011) has two phases, namely the Teaching phase and Learning phase. Through the teaching phase, learners get better from their teacher. The best population member here is the teacher in terms of both objectives and is used to bring learners (the remaining population) up to its level. The teaching phase expression is given by:

$$\boldsymbol{X_i^{t+1} = X_i^t + \left(r * \left(X_{mean} - \left(T_f \star X_{best}\right)\right)\right)} \qquad (\mathbf{3})$$

Where, $\boldsymbol{X_i^{t+1}}$ denotes the sequence of any member in the $(t+1)^{th}$ iteration, which is the immediate sequence after $\boldsymbol{X_i^t}$ (sequence of the members in $(t)^{th}$ iteration). $\boldsymbol{T_f}$ is termed the teaching factor (1 or 2) and $\boldsymbol{X_{mean}}$ is the mean sequence of all population members of any iteration, whose method is explained in the upcoming sections. The learning phase expression is given by:

$$\boldsymbol{X_i^{t+1} = X_i^{t+1} + \left(r * \left(X_i^t \pm \left(T_p * X_p^t\right)\right)\right)} \qquad (\mathbf{4})$$

Where, $\boldsymbol{X_p^t}$ denotes the partner population member picked randomly in order to undergo the learning phase and $\boldsymbol{T_p}$ is the partner factor.

The objective values are calculated for the strings in the population In every iteration, each population member first undergoes the teaching phase as shown in **equation (3)**, where the $\boldsymbol{X_{best}}$is the teacher, randomly picked from one corner solution of the front 1 found using NDS as explained earlier. The teachers phase updates each member in a partially induced manner determined by $\boldsymbol{T_f}$ & $\boldsymbol{r}$, and considers the collective position of every member determined by $\boldsymbol{X_{mean}}$. The same population member then undergoes a learning phase as shown in **equation (4)**, where it is compared with a randomly picked different partner solution and updated partially according to $\boldsymbol{T_p}$ & $\boldsymbol{r}$.

6.2.1 OPERATORS USED IN TLBO

The pseudocodes for the operators are presented below.

PSEUDOCODE FOR MEAN CALCULATION (Mean of All Task Sequences in the Population)

Y; *vector to hold the answer after carrying out the operation*
For *i in range(1, Number of tasks):*
    avg = 0; *variable to calculate average*

Xj; *task sequence[j'th member of the population]*
For *j in range(1, population size):*
avg = avg + Xj[i]
End for
avg = avg / number of tasks
If*(avg is already present in Y):*
Do:
R; *generate random number between [1 to number of tasks]*
While*(R is already present in Y)*
Y[i] = assign R
Else:
Y[i] = avg
End if
End for

PSEUDOCODE FOR SUBTRACTION OPERATOR (Task Sequence 1 - (Co-efficient * Task Sequence 2))

X1; *task sequence 1*
X2; *task sequence 2*
C; *co-efficient (Tf / Tp/ r)*
Y; *vector to hold the answer after carrying out the operation*
For *i in range(1, Number of tasks):*
If*(X1[i] is X2[i]):*
assign 0 to Y[i]
Else:
R; *generate random number between (0 to 1)*
If*(R < C):*
assign X2[i] to Y[i]
Else:
assign X1[i] to Y[i]
End if
End if
End for

PSEUDOCODE FOR ADDITION OPERATOR (Task Sequence1 + (Co-efficient * Task Sequence2))

X1; *task sequence 1*
X2; *task sequence 2*
C; *co-efficient (Tf / Tp/ r)*

```
Y; vector to hold the answer after carrying out the operation
For i in range(1, Number of tasks):
    If(X1[i] is X2[i]):
        assign X1[i] to Y[i]
    Else:
        R; generate random number between (0 to 1)
        If(R < C):
            assign X2[i] to Y[i]
        Else:
            assign 0 to Y[i]
        End if
    End if
End for
```

6.3 DISCRETE MIGRATING BIRDS OPTIMIZATION

Migrating Birds Optimization algorithm derives its analogy from the flocking of birds in a V-shaped manner during their collective flight. The analogy is such that the V-shape is the whole population wherein the tip of the V is assumed to contain the leader (or) the best solution. This algorithm undergoes three phases, Leader improvement, Block improvement and Leader updation phases (Duman et al., 2012)). The randomly picked corner solution from the pareto front-1 curve serves as the leader of the V shape.

In the leader improvement phase, the initial leader improves itself by randomly generating ‘k’ number of neighbor solutions or task sequences cum alternatives and compares itself amongst them using the same NDS procedure. The best solution amongst the ‘k+1’ solutions is assigned the leader. Next, for the block improvement phase, each of the remaining population members improve themselves by randomly generating ‘k-x’ neighbor solutions or task sequences cum alternatives each, amongst which they are compared for their individual best solutions and updated using the NDS approach. Lastly, the leader is updated by comparing all the newly updated population members using NDS. This procedure is repeated until the termination criteria of the algorithm is met.

6.3.1 NEIGHBORHOOD CREATION SCHEMES FOR MBO

Alongside, two neighborhood creation schemes for implementing discrete MBO are proposed in this paper, which are explained below.

PSEUDOCODE FOR NEIGHBORHOOD SOLUTION GENERATION-1 (By Changing Task Sequence Vector)

X; *task sequence vector*
Y; *vector to hold the answer after carrying out the operation*
R; *generate a random number between [1 to number of tasks]*
For *i in range (1, R):*
    Y[i] = X[i]
End for
For *i in range (R+1, number of tasks):*
    Do:
        q; *generate random number between [1 to number of tasks]*
    While*(q is already present in Y)*
    Y[i] = q
End for
Repair this task sequence Y ***(explained in section 6.5)***

PSEUDOCODE FOR NEIGHBORHOOD SOLUTION GENERATION-2 (By Changing Process Alternative Sequence Vector)

To change and find neighbor solutions of a particular solution, swapping and single point mutation is performed such that the constraints are satisfied.

- Swapping - randomly pick two indices from the sequence and swap their values such that the constraints are satisfied.
- Single point mutation - randomly pick an index and change its value randomly such that the constraints are satisfied.

For example for a 4 station problem,

Before changing - 1 3 1 2
After performing swap operation - 1 3 2 1 (last two positions are swapped)
After performing single point mutation - 2 3 2 1 (1st position is mutated)

6.4 MODIFIED DISCRETE ARCHIMEDES OPTIMIZATION ALGORITHM

The Archimedes optimization algorithm (AOA) 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 (population member) tries to reach equilibrium (optimized state). Each population member or task sequence cum process alternative has three values associated with it: Density, Volume and Acceleration (Hashim et al. 2021). The three values are updated for each task sequence and process alternatives until the

termination criteria is met as shown in **Algorithm 2.** The best solution among the population is found initially using the NDS approach and is the one with best density, volume and acceleration associated with it. First, for each remaining population, density and volume are updated as the iteration starts using the respective updation equations shown in **equation (5) & equation (6)**. The algorithm contains two parameters $\boldsymbol{T_f}$ & $\boldsymbol{d}$, being the transfer factor and density decreasing factor respectively, which increase gradually from 0 to 1 as they get updated in every iteration as shown in their respective updation equations **equation (7) & equation (8)**. The acceleration value followed by the task sequence cum process alternative of each population is then conditionally updated depending on the values of $\boldsymbol{T_f}$ and $\boldsymbol{d}$ as depicted in the below algorithm. These factors are based on the current and total iteration numbers, $\boldsymbol{t}$ and $\boldsymbol{t_{max}}$ respectively and play a vital role in bringing the whole population towards equilibrium, such that the algorithm has equal exploitation capability (when $\boldsymbol{T_f \leq 0.5}$, acceleration updation follows other random population ($\boldsymbol{X_r^t}$) as shown in **equation (9)**) and exploitation capacity (when $\boldsymbol{T_f > 0.5}$, acceleration updation follows current best solution ($\boldsymbol{X_{best}}$) as shown in **equation (10)**). The updated acceleration values are normalized as given in **equation (11)** after which the population/sequence is updated using **equation (12) & equation (13)**.

$$\boldsymbol{Den_i^{t+1} = Den_i^t + r * (Den_{best} - Den_i^t)} \quad (5)$$

$$\boldsymbol{Vol_i^{t+1} = Vol_i^t + r * (Vol_{best} - Vol_i^t)} \quad (6)$$

$$\boldsymbol{T_i^t = e^{((t - t_{max}) \div t_{max})}} \quad (7)$$

$$\boldsymbol{d^{t+1} = e^{((t_{max} - t) \div t_{max})} - (t \div t_{max})} \quad (8)$$

$$\boldsymbol{Acc_i = (Den_r + Vol_r * Acc_r) \div (Den_i - Vol_i)} \quad \boldsymbol{(if\ T_f \leq 0.5)} \quad (9)$$

$$\boldsymbol{Acc_i = (Den_{best} + Vol_{best} * Acc_{best}) \div (Den_i - Vol_i)} \quad \boldsymbol{(if\ T_f > 0.5)} \quad (10)$$

$$\boldsymbol{Acc_{i,NORM} = \left(0.9 * (Acc_i - Acc_{min}) \div (Acc_{max} - Acc_{min})\right) + 0.1} \quad (11)$$

$$\boldsymbol{X_i^{t+1} = X_i^t + \left(d * Acc_i^{t+1} * (X_r^t - X_i^t)\right)} \quad \boldsymbol{(if\ T_f \leq 0.5)} \quad (12)$$

$$\boldsymbol{X_i^{t+1} = X_i^t + \left(F * d * Acc_i^{t+1} * (X_{best} - X_i^t)\right)} \quad \boldsymbol{(if\ T_f > 0.5)} \quad (13)$$

For the specific purpose of solving and optimizing the HRC assembly line balancing problem and its considerations in this study, the sequence updating procedures for the AOA algorithm are modified as opposed to the standard procedures introduced by Hashim et al. (2021). This is done

so by removing the various randomization parameters that would otherwise contribute to excess computation. However, empirical trials were carried out to find the optimal task sequence before considering removing user-defined parameters. This gives an advantage to the AOA algorithm, that it requires no manual parameter fine tuning and the only parameters $\boldsymbol{T_f}$ and $\boldsymbol{d}$ are auto updated as the iterations progress. As discussed later in section 8**,** the relative performance of the hybridized AOA algorithm overrides the hybrids of the remaining algorithms. It also yields a large front while undergoing NDS, indicating that the algorithm does splendidly on the exploration end as well.

Moreover, as the density, volume and acceleration are all continuous variables, the aforementioned equations are simply applied for those variables. For updating the task sequence, the addition, subtraction and multiplication operators are the same as those given for the discretization procedures of TLBO **(section 6.2.1)**.

ALGORITHM 2: Pseudocode for AOA

*Initialize **T;** termination criteria (number of iterations)*
*Initialize **P;** population size (population has P objects)*
Initialize a random population (task sequences and process alternatives)
Using the decoding method find cycle time and energy values for the population
***CT;** holds the cycle time of each student in the population*
***E;** holds energy consumption of each student in the population*
Initialize density, volume and acceleration for the population randomly
***D;** holds density value of each object in the population (ranges between 0 to 1)*
***V;** holds volume value of each object in the population (ranges between 0 to 1)*
***A;** holds acceleration value of each object in the population (ranges between 0 to 1)*
***TF;** transfer operator*
***d;** density decreasing factor*
***EA;** external archive to store the pareto optimal front obtained*
Using fast non-dominated sorting find front 1 solutions for the current population
Add all these solutions to the EA
***For** t=1 to T (termination criteria)*
    ***For** i=1 to P (population size)*
        %UPDATE DENSITY & VOLUME%
        ***X_best;** best solution in the population*
        Using fast non-dominated sorting find front 1 for the current population
        From the front 1 pick the two corner points and assign one point as X_best
        ***D_best;** holds density value corresponding to the X_best*
        ***V_best;** holds acceleration value corresponding to the X_best*
        ***Di_t;** density of i'th object in current iteration*
        ***Vi_t;** volume of i'th object in current iteration*
        ***Di_t+1;** holds density of i'th object for next iteration*

Vi_t+1; *holds volume of i'th object for next iteration*
r; *random number generated between (0 - 1)*
Di_t+1; *found using* ***equation (5)***
Vi_t+1; *found using* ***equation (6)***
%UPDATE TRANSFER FACTOR AND DENSITY DECREASING FACTOR%
TF; *calculated using* ***equation (7)***
d; *calculated using* ***equation (8)***
%UPDATE ACCELERATION%
Ai_t; *acceleration of i'th object in the current population*
Ai_t+1; *holds acceleration of i'th object for next iteration*
If*(TF <= 0.5):*
    Generate a random number r between (1 to population size)
    Ar_t; *acceleration of r'th object in the current iteration*
    Dr_t; *density of r'th object in the current iteration*
    Vr_t; *volume of r'th object in the current iteration*
    Ai_t+1; *found using* ***equation (9)***
Else:
    A_best; *holds acceleration value corresponding to the X_best*
    Ai_t+1; *found using* ***equation (10)***
End if
%NORMALIZE ACCELERATION%
A_min; *minimum acceleration value in the population*
A_max; *maximum acceleration value in the population*
Ai_t+1(normalized); *found using* ***equation (11)***
%UPDATE TASK SEQUENCE%
Xi_t; *task sequence of i'th object*
Xi_t+1; *holds task sequence of i'th object*
If*(TF <= 0.5):*
    Xi_t+1; *found using* ***equation (12)***
Else:
    Generate a random number ***F*** *(either -1 or +1)*
    Xi_t+1; *found using* ***equation (13)***
End if
%UPDATE PROCESS ALTERNATIVE%
Yi_t; *process alternative of i'th student*
Yi_t+1; *holds process alternative of i'th student*
X_best_pa; *process alternative corresponding to X_best*
Yi_t+1 = Yi_t + X_best_pa
%REPAIR UPDATED TASK SEQUENCE AND PROCESS ALTERNATIVE%
Modify Xi_t+1 such that the new sequence follows the precedence relation

(explained in section 6.5)

Modify Yi_t+1 such that the constraints are followed

End for

%UPDATE EXTERNAL ARCHIVE%

Using fast non-dominated sorting find front 1 for the current population

Add all these solutions to the EA

End for

%FINDING PARETO OPTIMAL FRONT%

Using fast non-dominated sorting for all the solutions in the external archive

Remove all solutions from EA that is not in front 1

print*(EA); EA now holds the pareto optimal front solutions*

6.5 REPAIR FUNCTION USED TO GET FEASIBLE STRINGS

In all the algorithms developed in this study, the updated task sequences and process alternatives obtained after undergoing the proposed discretization methods in every iteration, might not satisfy the precedence constraints. To repair these sequences, a repair function is formulated in this study and is implemented parallelly with the developed algorithms to make sure that feasible sequences are generated.

ALGORITHM 3: Pseudocode for Repair Function:

X; *task sequence vector to be repaired*

Y; *a vector to hold repaired task sequence*

Available; *a vector that has available tasks that can be allocated (follows precedence)*

Insert tasks that have no precedence tasks into available vector

n = 0; *keep track of index in vector X*

i = 0; *keep track of index in vector Y*

Do:

If*(n >= N[total number of tasks]):*

Select a task randomly from the Available

T; randomly selected task from available

Insert T into Y[i]

Add tasks that have T as its precedence in Available

Remove T from the Available

Increment i by 1

Else if*(X[n] in Available):*

Insert X[n] into Y[i]

Add tasks that have X[n] as its precedence in Available

Remove X[n] from the Available

Increment n by 1

Increment i by 1

Else:

Increment n by 1

End if

While*(i < N [total number of tasks])*

Print*(Y); Y contains repaired task sequence (follows precedence relation)*

7. Hybridization of proposed multi-objective algorithms

Apart from the four metaheuristic algorithms that are considered, this study proposes to hybridize the four standard algorithms by embedding them with the scout phase of the ABC algorithm.

In each iteration, the scout phase of the ABC keeps track of the individual population members or task cum process alternative sequences that are not improved/updated to a new objective value over the iterations (Karaboga and Basturk, 2007). **Algorithm 4** shows the sequential steps carried out, where a limit value is manually set for the allowed iteration count for which objective values (cycle time & energy consumption) of any population remain unchanged. Therefore, if the objective values of any population member remain unchanged over the iteration limit value that was set manually, this phase aids in the removal of that particular population member and replaces it with a random different population.

This mechanism of remote tracking, removal and replacement of members with repeating objective values prevents the algorithm from stagnating in local optima zones, enabling the population updation to converge better towards the narrowed set of pareto optimal values as each iterations progress. Moreover, the removed solution(s) may be the pareto optimal solution, for which purpose, they are stored/memorized separately in the EA using the elitist approach and are then compared at the algorithm termination point.

ALGORITHM 4: Pseudocode for Scout Phase of ABC Algorithm (used for Hybridization)

Counter; *a vector that keep tracks of no. of times a solution in the population is unchanged*

For *i=1 to P (population size)*

Counter_i = 0; *Counter for all the solutions in the population is set to 0 initially*

End for

Limit; *a value set by user*

For *t=1 to T (number of iterations)*

For *i=1 to P (population size)*

CTi_old; *holds cycle time value of i'th solution before updation*

Ei_old; *holds energy value of i'th solution before updation*

…

Run updation procedure using any algorithm

…
CTi_new; *holds cycle time value of i'th solution after updation*
Ei_new; *holds energy value of i'th solution after updation*
If*(CTi_old & Ei_old SAME AS CTi_new & Ei_new):*
Increment Counter_i by 1
End if
If*(Counter_i >= Limit):*
Remove present i'th solution from the population
Add a new randomly generated solution in that position
Set Counter_i to 0
End if
End for
End for

8. Experimental design and results

This section presents the results obtained by analytical plots and statistical comparison of both the hybridized algorithms along with their base algorithms using three novel performance indications that depict the all-round characteristics of any metaheuristic algorithms. Further, the proposed fine-tuning methodologies for each of the developed algorithms are given and their results as to the best fine-tuning criteria are discussed. Lastly, the developed algorithms are validated by a case-study implemented in an electronic assembly industry.

8.1 Experimental design

To address the lack of HRC datasets in the literature, thirty-two standard HRC datasets as shown in **table 3** are categorized based on the sizes of tasks and workstations for decoding purposes. Of the thirty-two datasets, problems are categorized on the basis of the number of tasks: small (number of tasks < 50); medium (number of tasks < 100); large (number of tasks >100), wherein the sizes of tasks ranges from 25 to 297, with varying workstation count. In this study, the approach followed in presenting the results of every algorithmic performance are based on these categories of task sizes, i.e., how each algorithm performs throughout these task size ranges and which algorithm has the highest significance at each task and workstation sizes of any HRC problem.

The detailed reference of the experimental setup that has been considered to carry out this study is given in the data generation methods under **section 4**. The quantity of manual workers are not constrained whereas the study considered 3 types of robots with unlimited availability of each type. The energy consumption of robots largely varies depending on the application and thus the energy ratings of the robots were randomly generated in the range [0.2, 0.4] kW. The energy rating

of humans translates relatively from wages and is empirically set as 0.2, which is lower than that of robots. The operational and other associated times are randomly generated as shown in the data generation methodology, using upper and lower bound values to account for deviations.

Table 3. Standard testing dataset premise used for comparison of algorithms

Problem number	No. of Tasks	No. of stations	Problem number	No. of Tasks	No. of stations
1	25	3	**17**	89	8
2	25	4	**18**	89	12
3	25	6	**19**	89	16
4	25	9	**20**	89	21
5	35	4	**21**	111	9
6	35	5	**22**	111	13
7	35	7	**23**	111	17
8	35	12	**24**	111	22
9	53	5	**25**	148	10
10	53	7	**26**	148	14
11	53	10	**27**	148	21
12	53	14	**28**	148	29
13	70	7	**29**	297	19
14	70	10	**30**	297	29
15	70	14	**31**	297	38
16	70	19	**32**	297	50

It is assumed that the task times of cobots (worker + robot) are the lowest and for the robots are the highest. The considerations for population size and iteration count were empirically determined and were taken as thirty each, for all developed standard & hybrid algorithms. The suitability of

all the developed hybridized algorithms and their base algorithms are compared using the three performance indicators discussed in section 8.2. The average objective function values of ten runs of each algorithm is considered for evaluation.

8.2. Performance Indicators

Three performance indicators are adopted from Rashid et al. (Ab Rashid, Hutabarat, and Tiwari 2018) to conduct the comparative studies, which are detailed below.

Perfromance Indicator-1: ***Non-dominated solutions count*** (N), which denotes the number of non-dominated values found in the Pareto optimal set after the last iteration is complete. A higher value of (N) denotes better algorithmic performance.

Perfromance Indicator-2: ***Global error ratio*** (E_G), which denotes the percentage of the non-dominated solution count (N) to the global non-dominated value count (Pareto optimal set obtained from combining the individual pareto optimal sets of all algorithms under comparison). Hence, a lower value of (E_G) has an indirect relative correlation with (N) and signifies good performance of the algorithm.

Perfromance Indicator-3: ***Convergence metric*** (CM), which gives a measure of the extent of convergence towards the global Pareto optimal set. Hence, a lower value of CM depicts better algorithmic performance. It is mathematically determined by calculating the sum of euclidean distances d_i between the nearest end iteration values of non-dominated solutions and the global Pareto optimal set values and dividing by N.

$$CM = \frac{\sum_{i=1}^{N} d_i}{N} \quad \textbf{(14)}$$

8.3 Parametric Fine tuning

Every metaheuristic algorithm has certain parameters embedded, which aids to bias the search process in converging every solution towards an optimal value with different considerations. To aid in reducing the complexity of fine tuning of the algorithmic parameters, this study proposes a new range methodology to determine the best range for all the parameters involved in the four base algorithms considered, based on the size of the problems. Depending on the influence of parameters in the equation (equation 1 to 13), the different parametric value ranges for all four base algorithms are given as shown in **table 4**. Due to the page restriction, the complete results of sensitivity analysis are not presented. While running the code for the algorithms, parameter values are generated randomly within the range identified and presented in column three of table 4.

Table 4. Proposed parameter fine tuning results based on ranges

ALGORITHM	RANGES	IDENTIFIED PARAMETER RANGE FOR DIFFERENT PROBLEM SIZES AFTER SENSITIVITY ANALYSIS CONDUCTED
PSO (**c1**-inertia weight **c2**-acceleration coefficient of P_best **c3**-acceleration coefficient of G_best)	• **Range 1 :** c1 ∈ (0, 0.5), c2+c3 <= 1.5 • **Range 2 :** c1 ∈ (0.5, 1), c2+c3 <= 1.5 • **Range 3 :** c2 ∈ (0, 0.5), c1+c3 <= 1.5 • **Range 4 :** c2 ∈ (0.5, 1), c1+c3 <= 1.5 • **Range 5 :** c3 ∈ (0, 0.5), c1+c2 <= 1.5 • **Range 6 :** c3 ∈ (0.5, 1), c1+c2 <= 1.5	**SMALL SIZE PROBLEMS - Range 1** **MEDIUM SIZE PROBLEMS - Range 6** **LARGE SIZE PROBLEMS - Range 4**
TLBO (**r**-mean impact coefficient **Tf**-Teaching coefficient **Tp**-Learning coefficient)	• **Range 1 :** r ∈ (0, 0.5), Tf+Tp <= 1.5 • **Range 2 :** r ∈ (0.5, 1), Tf+Tp <= 1.5 • **Range 3 :** Tf ∈ (0, 0.5), r+Tp <= 1.5 • **Range 4 :** Tf ∈ (0.5, 1), r+Tp <= 1.5 • **Range 5 :** Tp ∈ (0, 0.5), r+Tf <= 1.5 • **Range 6 :** Tp ∈ (0.5, 1), r+Tf <= 1.5	**SMALL SIZE PROBLEMS - Range 3** **MEDIUM SIZE PROBLEMS - Range 6** **LARGE SIZE PROBLEMS - Range 3**
MBO (**K** - number of neighborhood solutions of current leader **X** - represents any number less than K)	• **Range 1 :** K=2, X=1 • **Range 2 :** K=3, X=1 • **Range 3 :** K=5, X=2 • **Range 4 :** K=5, X=3 • **Range 5 :** K=7, X=3 • **Range 6 :** K=10, X=5	**SMALL SIZE PROBLEMS - Range 4** **MEDIUM SIZE PROBLEMS - Range 2** **LARGE SIZE PROBLEMS - Range 3**
AOA	No fine tuning needed as parameters are neglected for the developed & improved AOA.	----------

Using the performance indicators listed in **section 8.2**, the performance of each algorithm for all its associated six ranges are compared for the small, medium and large size problems. The best parametric range is decided for each algorithm based on the size of the problem.

8.4 Comparative results

All eight algorithms have been evaluated using the three performance indicators discussed in section 8.2. For instance, **figure 5** shows the comparison plots of both standard & hybridized versions of PSO, TLBO, MBO and AOA using the performance indicator-1 (N) (on y-axis) for the thirty-two datasets (on x-axis). It can be observed that AOA shows higher peaks indicating a greater performance compared to the rest. Plots are generated for the other performance indicators as shown in **figure 6 and 7**. The AOA algorithm manifests a better performance with a lower error ratio and convergence metric. The reader could however compare any relative algorithmic performances based on the different task sizes by simply navigating through the abscissa (task sizes) and ordinate (performance values) of the plots.

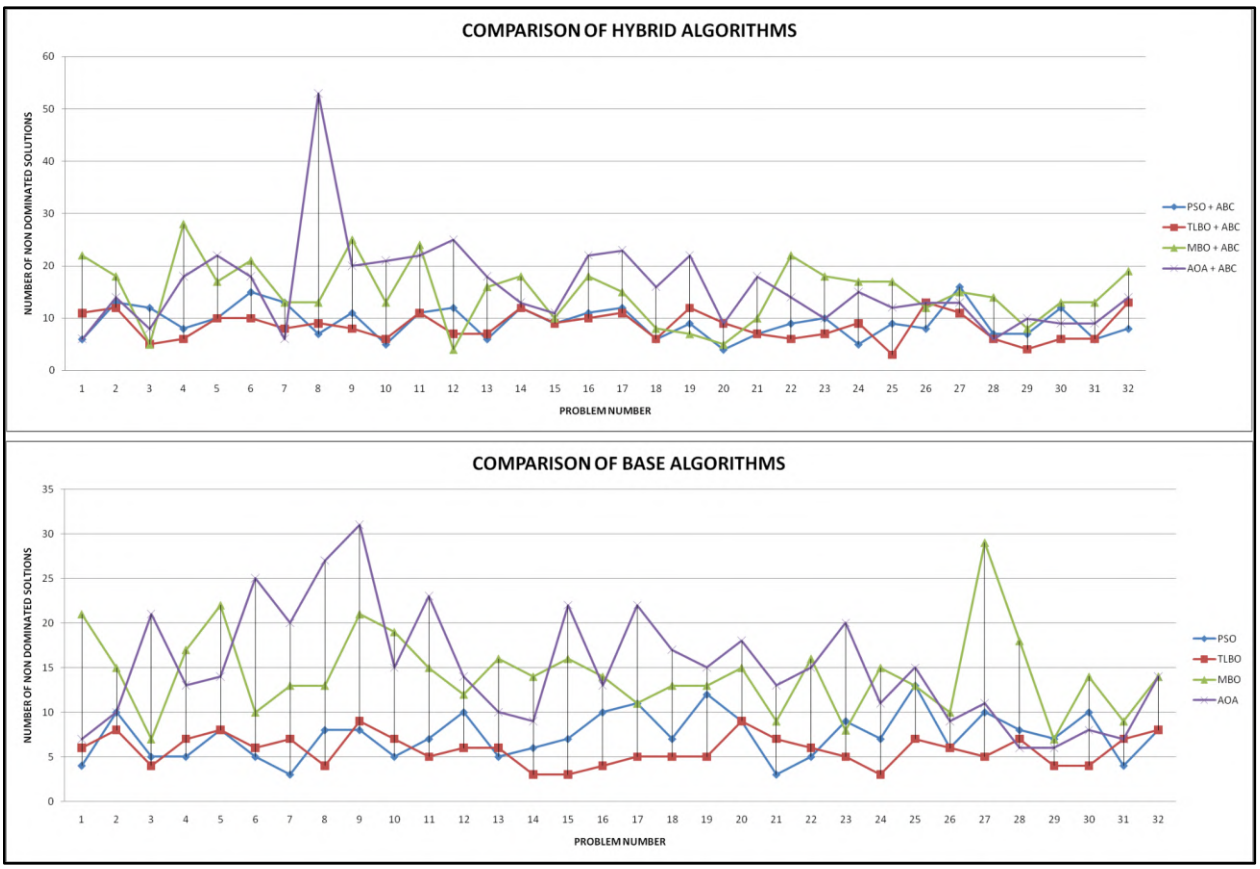

Figure 5. Line plot of Performance Indicator 1 (N)

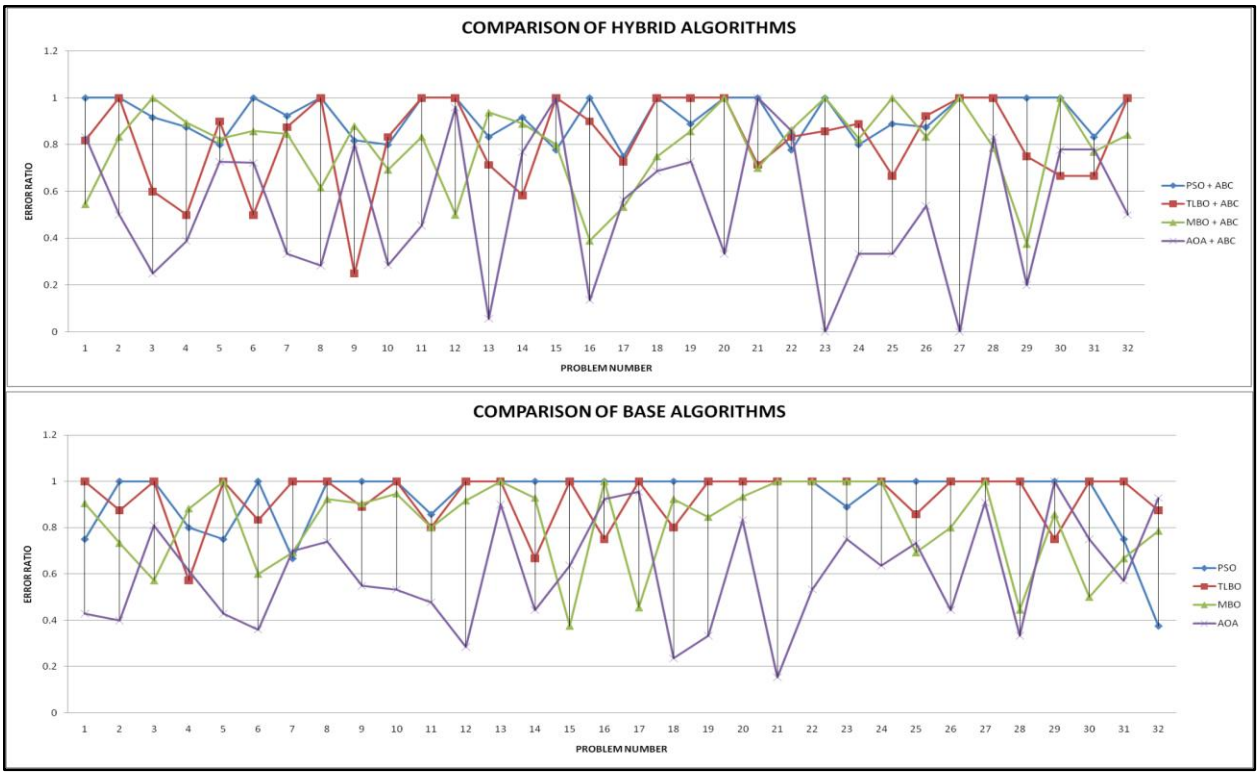

Figure 6. Line plot of Performance Indicator 2 (E_G)

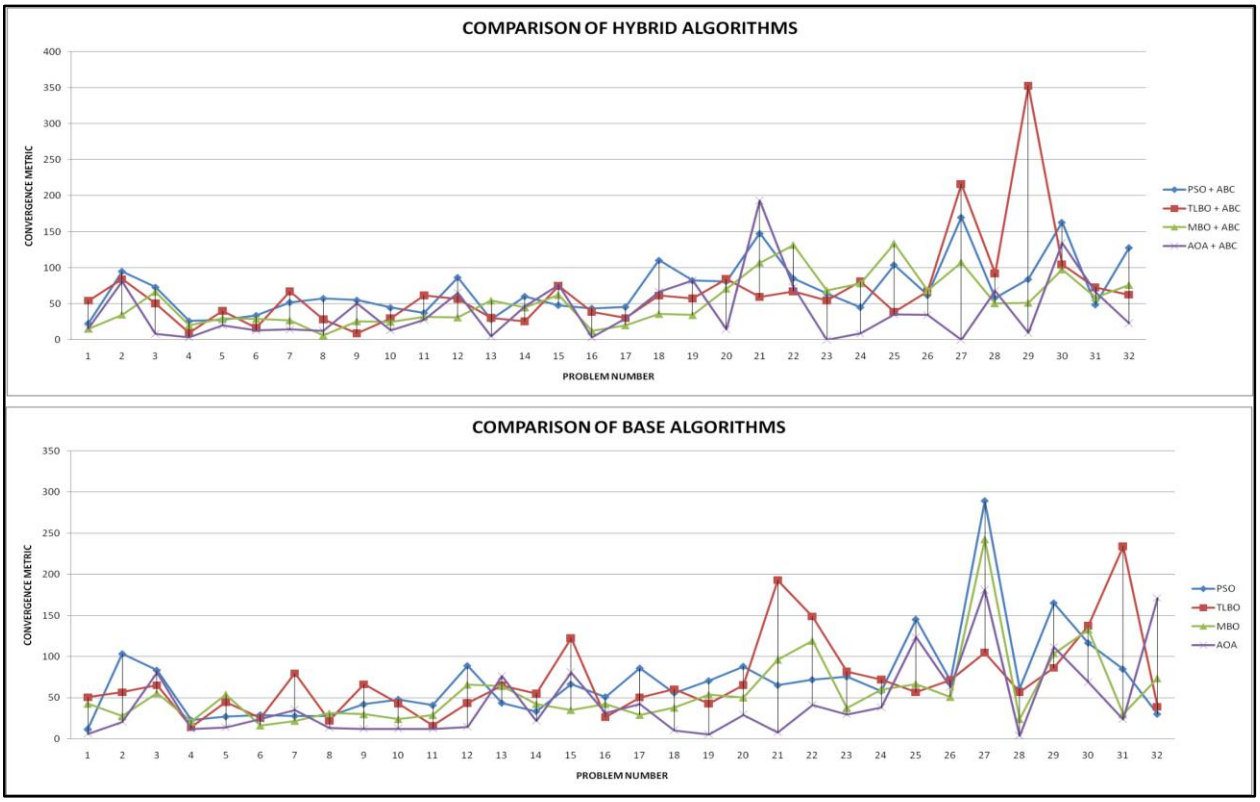

Figure 7. Line plot of Performance Indicator 3 (CM)

Apart from the hybridized and base algorithms of both AOA and MBO, the performance of all other algorithms have almost performed similarly for all the thirty-two datasets. Due to the space restriction, other results are not included in the paper.

8.5 Statistical Analysis

To get numerical estimates and confirmation on the performance of the algorithms presented in section 8.4, statistical analysis is carried out using Friedman's test. The result of the test is shown in **table 5**. The test confirms that the eight algorithms are significantly different for two out of three performance indicators with p-value < 0.05, which could be observed from the plots presented in section 8.4.

Table 5. Friedman's Test results for base & hybrid algorithms

ALGORITHM TYPE	**PERFORMANCE INDICATOR**	Number of data sets **(n)**	Chi-square value **(χ2)**	Degrees of freedom **(dof)**	**p- value** (significance indicator)
BASE	Non dominated solutions count (N)	32	61.434	3	< 0.00001
	Global error ration (E_G)	32	29.025	3	0.00028
	Convergence metric (CM)	32	23.888	3	0.287
HYBRID	Non dominated solutions count (N)	32	39.722	3	< 0.00001
	Global error ration (E_G)	32	18.35	3	0.00037
	Convergence metric (CM)	32	18.488	3	0.343

Given the existence of significant differences among the algorithms for two performance indicators (N) and (E_G), Post-hoc test (Scheffé multiple comparison test) is conducted for comparing the relative performance of all individual algorithms, to test their significance. From the post-hoc results carried out as shown in **table 6** for performance indicator (N), it could be seen that the AOA algorithm and its hybrid shows the most significance when compared with the rest of the algorithms. The underlined treatment pair cells shown in the results table depict the comparisons that involve AOA. As this study categorizes the thirty-two task datasets into sizes of small, medium and large, it can also be concluded that the standard and hybridized AOA algorithm

performs better than the rest of the algorithms for all problem categories. Similarly, **table 6** also shows the post-hoc test results carried out for indicator (E_G), which furthermore shows the highest significance count of the hybridized AOA algorithm. To extend the grounds of algorithmic performance, the pareto optimal fronts obtained through NDS by all the developed algorithms after the termination criteria of 30 iterations are plotted to visualize the relative and collective positions of the multi-objective optimized solutions. From **figure 8,** the pareto fronts obtained by the hybridized AOA algorithm yields a relatively more spread front signifying the increased exploration capacity due to addition of the scout phase mechanism. The pareto front of hybrid AOA (figure 8) is also the closest to the origin of the two-dimensional objective space signifies the better performance. The results of post-hoc analysis presented in table 6 shows that the pairs with AOA or its hybrid combination explains the significant difference when compared with any other algorithm with $p < 0.01$.

Table 6. Scheffé multiple comparison test results using non-dominated solutions count(N) & global error ratio (E_G)

PERFORMANCE INDICATORS	**Treatments pair**	**Scheffé T-statistic**	**Scheffé p-value**	**Scheffé inference**
N	PSO vs PSO+ABC	1.5206	0.9397136	insignificant
E_G	PSO vs PSO+ABC	0.2398	0.9999996	insignificant
N	PSO vs TLBO	1.2215	0.9822245	insignificant
E_G	PSO vs TLBO	0.1123	1.0000000	insignificant
N	PSO vs TLBO+ABC	0.8725	0.9977599	insignificant
E_G	PSO vs TLBO+ABC	2.4252	0.5548421	insignificant
N	PSO vs MBO	5.5839	0.0001095	** p<0.01
E_G	PSO vs MBO	2.4807	0.5235409	insignificant
N	PSO vs MBO+ABC	6.0575	1.3529e-05	** p<0.01
E_G	PSO vs MBO+ABC	2.8873	0.3084131	insignificant
N	PSO vs AOA	6.1323	9.5683e-06	** p<0.01
E_G	PSO vs AOA	6.9436	1.7142e-07	** p<0.01
N	PSO vs AOA+ABC	6.8552	2.7179e-07	** p<0.01

E_G	PSO vs AOA+ABC	8.5068	2.2587e-11	** p<0.01
N	PSO+ABC vs TLBO	2.7421	0.3805235	insignificant
E_G	PSO+ABC vs TLBO	0.1275	1.0000000	insignificant
N	PSO+ABC vs TLBO+ABC	0.6481	0.9996801	insignificant
E_G	PSO+ABC vs TLBO+ABC	2.1854	0.6870097	insignificant
N	PSO+ABC vs MBO	4.0633	0.0238375	* p<0.05
E_G	PSO+ABC vs MBO	2.2409	0.6573650	insignificant
N	PSO+ABC vs MBO+ABC	4.5369	0.0056326	** p<0.01
E_G	PSO+ABC vs MBO+ABC	2.6475	0.4308202	insignificant
N	PSO+ABC vs AOA	4.6117	0.0043954	** p<0.01
E_G	PSO+ABC vs AOA	6.7038	5.9074e-07	** p<0.01
N	PSO+ABC vs AOA+ABC	5.3346	0.0003057	** p<0.01
E_G	PSO+ABC vs AOA+ABC	8.2670	9.7203e-11	** p<0.01
N	TLBO vs TLBO+ABC	2.0939	0.7339294	insignificant
E_G	TLBO vs TLBO+ABC	2.3130	0.6178117	insignificant
N	TLBO vs MBO	6.8053	3.5161e-07	** p<0.01
E_G	TLBO vs MBO	2.3684	0.5868431	insignificant
N	TLBO vs MBO+ABC	7.2790	2.8504e-08	** p<0.01
E_G	TLBO vs MBO+ABC	2.7750	0.3635503	insignificant
N	TLBO vs AOA	7.3537	1.8919e-08	** p<0.01
E_G	TLBO vs AOA	6.8314	3.0742e-07	** p<0.01

N	TLBO vs AOA+ABC	8.0766	3.0302e-10	** p<0.01
E_G	TLBO vs AOA+ABC	8.3945	4.4892e-11	** p<0.01
N	TLBO+ABC vs MBO	4.7114	0.0031316	** p<0.01
E_G	TLBO+ABC vs MBO	0.0554	1.0000000	insignificant
N	TLBO+ABC vs MBO+ABC	5.1850	0.0005516	** p<0.01
E_G	TLBO+ABC vs MBO+ABC	0.4621	0.9999675	insignificant
N	TLBO+ABC vs AOA	5.2598	0.0004116	** p<0.01
E_G	TLBO+ABC vs AOA	4.5184	0.0059837	** p<0.01
N	TLBO+ABC vs AOA+ABC	5.9827	1.9047e-05	** p<0.01
E_G	TLBO+ABC vs AOA+ABC	6.0815	1.2109e-05	** p<0.01
N	MBO vs MBO+ABC	0.4736	0.9999615	insignificant
E_G	MBO vs MBO+ABC	0.4066	0.9999865	insignificant
N	MBO vs AOA	0.5484	0.9998958	insignificant
E_G	MBO vs AOA	4.4629	0.0071595	** p<0.01
N	MBO vs AOA+ABC	1.2713	0.9775765	insignificant
E_G	MBO vs AOA+ABC	6.0261	1.5628e-05	** p<0.01
N	MBO+ABC vs AOA	0.0748	1.0000000	insignificant
E_G	MBO+ABC vs AOA	4.0563	0.0243050	* p<0.05
N	AOA vs AOA+ABC	0.7229	0.9993403	insignificant
E_G	AOA vs AOA+ABC	1.5631	0.9303866	insignificant
N	MBO+ABC vs AOA+ABC	0.7977	0.9987435	insignificant

E_G	MBO+ABC vs AOA+ABC	5.6194	9.4186e-05	** p<0.01

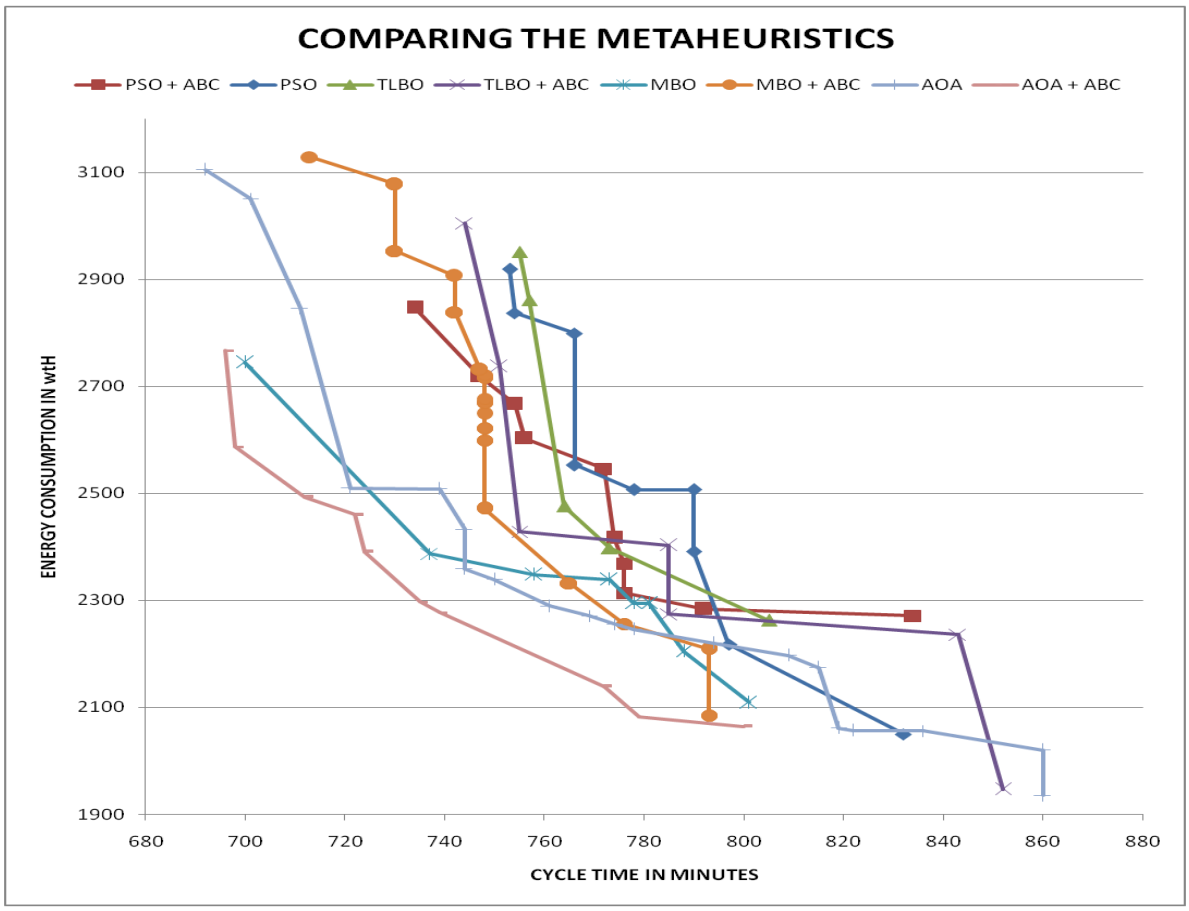

Figure 8. End iteration Pareto optimal fronts obtained by all 8 algorithms

8.6 Case-study

With reference to the set of hybridized algorithms developed in this study along with the different data generation methods and flexible resource constraint allocation capabilities, the proposed methodology for HRC assembly line is supported by a case study done in an electronics assembly industry. This is a semi-automated assembly system that assembles PCB components.

8.6.1 Line description & modeling

The current scenario of the considered sub-assembly process are as follows: All tasks in the observed sub-assembly segment were carried out manually (study aims to semi-automate existing manual operations). The workstation count is subjected to shop-floor space availability and is considered to be from 2 to 5. The housing assembly consisted of 23 tasks pertaining to task characteristics like shifting, placing & positioning, fastening, intermediate torque, quality & part inspections, and intricate wire positioning (only done manually). Before testing the HRC line balancing performance by the developed metaheuristics, the primary objective involved converting the existing manual sub-assembly process into a "semi-automated" one by considering managerial cost and automation resource implications.

The methodology followed to carry the study in the chosen collaborative assembly system are as follows:

- All tasks are observed and documented of their operation times, their possible implementation methods and task precedence relations.
- The best possible automation to be introduced is discussed with the IE & automation teams of the industry for their specifications and suitability in assembling sensitive electronic components. After careful analysis of all the recorded tasks, it was decided that two types of robots (R1 and R2), one with mechanical grippers to perform pick & place operations and another with end-effectors of suction tube for delicate portions.
- The task constraints of all the available resources were formulated after ergonomic and safety considerations.
- From various industrial robot specification sources, the average energy ratings of both the robotic types (6-axis-7kg(max)-articulated automatic numerical control robots) are taken to be in the range of [0.6-0.8] kW.
- The setup times measured for both types of robots for all tasks were observed to be very insignificant w.r.t the operational times. Hence, it can be assumed for this study that the operational time bounds would have accounted for the individual setup times.
- Since the assembly is highly compact and integrated, the resource allocations are optimally spread to avoid the possibility of collisions.
- The availability of human workers is assumed unconstrained.

The modeled HRC sub-assembly inputs of 23 tasks finally need to be optimally allocated among the possible number of workstations by assigning the available manual and given collaborative robot resources with the aim to balance the line by reducing total cycle time & line energy consumption. It should be noted that due to confidential reasons in line with the company's legal policies, only the outline data of the sub-assembly process has been listed.

8.6.2 Performance analysis and results discussion

After running the hybridized AOA algorithm for the given modeled data, the resulting cycle time and energy consumption values for different workstation counts are first compared between the existing manual assembly line to that of the generated HRC assembly line. Thus, the comparison shown in **figure 9** provides the necessary insights for deciding the right workstation count by comparing the simultaneous relative decrease in cycle times and increase in energy consumption for every workstation count (represented in the abscissa). Through the multi-objective implementations carried out, a workstation count of '3' shows the largest decrease in cycle time observed for a relatively less hike in line energy consumption. Although it is certain that energy consumption is increased when automation is added, the long term productivity is increased which compensates for the relatively small increase in energy consumption.

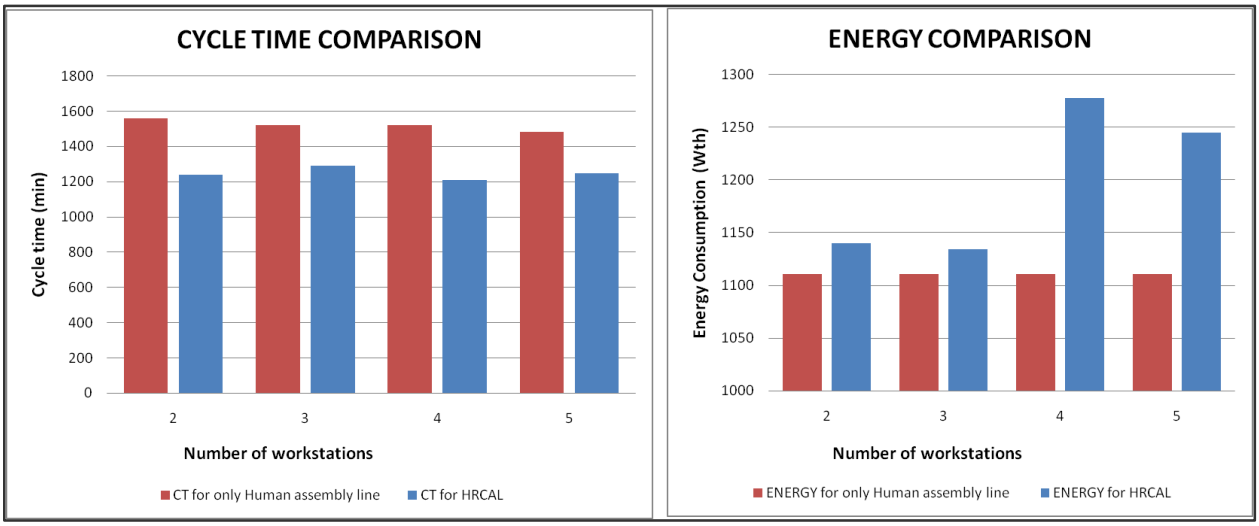

Figure 9. Insights from case study

Having chosen three workstations as the optimal shop-floor space utilization criteria, the hybrid AOA algorithm is run for the said workstation count and the input models depicted earlier. **Figure 10** shows the HRC allocation that has been obtained by AOA with the best task sequence solution among the generated pareto optimal set. According to this allocation, the first workstation is assigned a manual space, the second with a collaborative space of robot R1, and the third with yet another collaborative space of robot R2, all with their best possible respective tasks. Through this case-study implementation, the use of metaheuristics in solving line balancing problems proves to be largely flexible in arriving at the best optimal sequences, considering several resource and management constraints. In practical situations, high volume projects in an industry site the need to automate most of their processes effectively. With the available level of current automation and the long term anticipated growth of unfolding multiple projects, managers can extend such HRC studies into manual lines that are critical to the throughput of a product.

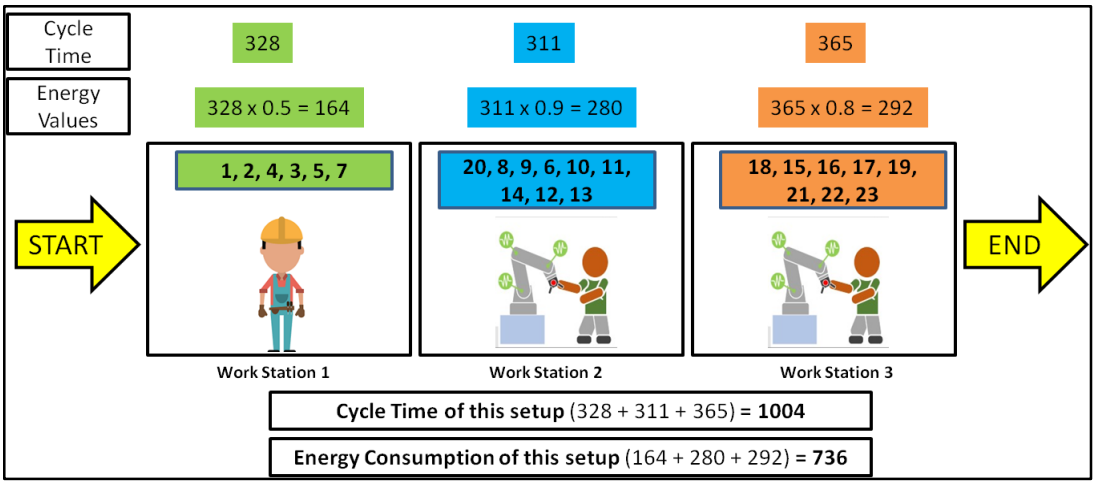

Figure 9. Optimal HRC allocation generated based on the collaborative model

9. Conclusion and future research directions

The eight developed metaheuristic algorithms are used to balance the HRC assembly line that minimizes the cycle time energy consumption. A novel data generating methodology to generate task times, setup and sequence dependent setup times have been proposed. The proposed fine-tuning methodologies and discretization procedures in this study for all the algorithms can be aligned as a benchmark towards further easy implementation of future studies that employ similar metaheuristics to solve discrete HRC assembly line problems.

Having identified the lack of standard testing HRC datasets, a set of thirty-two datasets are generated, and they are grouped as small, medium and large size problems. These problems are used to evaluate the eight metaheuristic algorithms. Three performance indicators namely, number of nondominated solutions (N), global error ratio (E_G) and convergence metric (CM) are used as the basis for evaluation. The statistical analysis conducted reveals that the eight algorithms are significantly different for two out of three performance indicators with p-value < 0.05. The statistical analysis gave another insight that the hybridized and standard AOA algorithm showed more than 50% better performance when compared to the others. This test highlights the potential of AOA in being used as an ideal metaheuristic for solving discrete ALB and assembly sequence planning, which can also be supported by the large and near-to-origin optimal pareto fronts obtained for AOA. Throughout the study, the pseudocodes of every proposed framework are sequentially explained to save time in interpretation for new researchers. The study also sheds light on task sequence repair functions, whose associated problems are usually faced by researchers in coding discrete applications.

Though the study of metaheuristic performances and line balancing for HRC compliances considers a set of assumptions, these instances would prove particularly useful when applied for a real-world formulated case study in an appropriate industry. A case study had been conducted in

the electronic assembly system. The primary objective of conducting a case study is to introduce a certain amount of automation for the current traditional methods involved. The best performing AOA had been used to generate the results. The case study conducted suggests the number of workstations that minimizes cycle time and energy consumption. The line balancing results obtained in this study for the manual turned semi-automated electronic sub-assembly process showed the desired results of lowering the cycle time with considerable energy consumption of automation resources. It also proves the viability of using metaheuristic algorithms, developing their performance and then running a vast multitude of realistic constraints to produce the best set of optimal assembly line allocation results.

For research on HRC line balancing in the coming times, many concepts related to production planning and control methods could be adopted. For instance, the layout alternatives for different HRC settings as listed briefly by Qiuhua et al. (Tang et al. 2017) can be looked at much deeper. As depicted in this study, the developed metaheuristic algorithms can be implemented in real-case industrial problems as most industries provide the flexibility of several product assembly lines and the research on HRC could be aligned with real world industrial considerations rather than pure assumptions.

REFERENCES

Ab Rashid, Mohd Fadzil Faisae, Windo Hutabarat, and Ashutosh Tiwari. 2018. "Multi-Objective Discrete Particle Swarm Optimisation Algorithm for Integrated Assembly Sequence Planning and Assembly Line Balancing." *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 232 (8). SAGE Publications Ltd: 1444–1459. doi:10.1177/0954405416673095.

Berx, Nicole, Wilm Decré, Ido Morag, Peter Chemweno, and Liliane Pintelon. 2022. "Identification and Classification of Risk Factors for Human-Robot Collaboration from a System-Wide Perspective." *Computers and Industrial Engineering* 163 (January). Elsevier Ltd. doi:10.1016/j.cie.2021.107827.

Borba, Leonardo, Marcus Ritt, and Cristóbal Miralles. 2018. "Exact and Heuristic Methods for Solving the Robotic Assembly Line Balancing Problem." *European Journal of Operational Research* 270 (1): 146–156. doi:10.1016/j.ejor.2018.03.011.

Boysen, Nils, Philipp Schulze, and Armin Scholl. 2021. "Assembly Line Balancing: What Happened in the Last Fifteen Years?" *European Journal of Operational Research*. Elsevier B.V. doi:10.1016/j.ejor.2021.11.043.

Chutima, Parames. 2022. "A Comprehensive Review of Robotic Assembly Line Balancing Problem." *Journal of Intelligent Manufacturing*. Springer. doi:10.1007/s10845-020-01641-7.

Çil, Zeynel Abidin, Zixiang Li, Suleyman Mete, and Eren Özceylan. 2020. "Mathematical Model and Bee Algorithms for Mixed-Model Assembly Line Balancing Problem with Physical Human–Robot Collaboration." *Applied Soft Computing Journal* 93 (August). Elsevier Ltd. doi:10.1016/j.asoc.2020.106394.

Dalle Mura, Michela, and Gino Dini. 2019. "Designing Assembly Lines with Humans and Collaborative Robots: A Genetic Approach." *CIRP Annals* 68 (1). CIRP: 1–4. doi:10.1016/j.cirp.2019.04.006.

Dalle Mura, Michela, and Gino Dini. 2022. "Job Rotation and Human–Robot Collaboration for Enhancing Ergonomics in Assembly Lines by a Genetic Algorithm." *International Journal of Advanced Manufacturing Technology* 118 (9–10). Springer London: 2901–2914. doi:10.1007/s00170-021-08068-1.

Deb, Kalyanmoy, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. 2002. "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II." *IEEE Transactions on Evolutionary Computation* 6 (2): 182–197. doi:10.1109/4235.996017.

Dokeroglu, Tansel, Ender Sevinc, Tayfun Kucukyilmaz, and Ahmet Cosar. 2019. "A Survey on New Generation Metaheuristic Algorithms." *Computers and Industrial Engineering* 137 (November). Elsevier Ltd. doi:10.1016/j.cie.2019.106040.

Duman, E., Uysal, M. and Alkaya, A.F., 2012. Migrating birds optimization: a new metaheuristic approach and its performance on quadratic assignment problem. Information Sciences, 217, pp.65-77.

Gualtieri, Luca, Erwin Rauch, and Renato Vidoni. 2021. "Methodology for the Definition of the Optimal Assembly Cycle and Calculation of the Optimized Assembly Cycle Time in Human-Robot Collaborative Assembly." *International Journal of Advanced Manufacturing Technology* 113 (7–8). The International Journal of Advanced Manufacturing Technology: 2369–2384. doi:10.1007/s00170-021-06653-y.

Hamzadayi, Alper. 2018. "Balancing of Mixed-Model Two-Sided Assembly Lines Using Teaching–Learning Based Optimization Algorithm." *Pamukkale University Journal of Engineering Sciences* 24 (4). LookUs Bilisim, Ltd.: 682–691. doi:10.5505/pajes.2017.14227.

Hashim, Fatma A., Kashif Hussain, Essam H. Houssein, Mai S. Mabrouk, and Walid Al-Atabany. 2021. "Archimedes Optimization Algorithm: A New Metaheuristic Algorithm for Solving Optimization Problems." *Applied Intelligence* 51 (3). Springer: 1531–1551. doi:10.1007/s10489-020-01893-z.

Karaboga D., Basturk B. (2007), Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems, Advances in Soft Computing: Foundations of Fuzzy Logic and Soft Computing, , volume 4529/2007 of LNCS: 789-798, Springer, Berlin.

Kashani, Ali R., Raymond Chiong, Seyedali Mirjalili, and Amir H. Gandomi. 2021. "Particle Swarm Optimization Variants for Solving Geotechnical Problems: Review and Comparative Analysis." *Archives*

of Computational Methods in Engineering 28 (3). Springer Science and Business Media B.V.: 1871–1927. doi:10.1007/s11831-020-09442-0.

Kennedy, J., and R. Eberhart. 1995. Particle swarm optimization." In 1995 IEEE International Conference on Neural Networks Proceedings", vol. 1, p. 6. 1948.

Koltai, Tamás, Imre Dimény, Viola Gallina, Alexander Gaal, and Chiara Sepe. 2021. "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* 242 (December). Elsevier B.V. doi:10.1016/j.ijpe.2021.108292.

Levitin, G., Rubinovitz, J. and Shnits, B., 2006. A genetic algorithm for robotic assembly line balancing. European Journal of Operational Research, 168(3), pp.811-825.

Li, Zixiang, Mukund Nilakantan Janardhanan, and S. G. Ponnambalam. 2021. "Cost-Oriented Robotic Assembly Line Balancing Problem with Setup Times: Multi-Objective Algorithms." *Journal of Intelligent Manufacturing* 32 (4). Springer: 989–1007. doi:10.1007/s10845-020-01598-7.

Li, Zixiang, Mukund Nilakantan Janardhanan, and Qiuhua Tang. 2021. "Multi-Objective Migrating Bird Optimization Algorithm for Cost-Oriented Assembly Line Balancing Problem with Collaborative Robots." *Neural Computing and Applications*. Springer Science and Business Media Deutschland GmbH. doi:10.1007/s00521-020-05610-2.

Li, Z., Janardhanan, M.N., Tang, Q. and Ponnambalam, S.G., 2019. Model and metaheuristics for robotic two-sided assembly line balancing problems with setup times. Swarm and Evolutionary Computation, 50, p.100567.

Malik, Ali Ahmad, and Arne Bilberg. 2019. "Complexity-Based Task Allocation in Human-Robot Collaborative Assembly." *Industrial Robot* 46 (4): 471–480. doi:10.1108/IR-11-2018-0231.

Nourmohammadi, Amir, Masood Fathi, and Amos H.C. Ng. 2022. "Balancing and Scheduling Assembly Lines with Human-Robot Collaboration Tasks." *Computers and Operations Research* 140 (April). Elsevier Ltd. doi:10.1016/j.cor.2021.105674.

Parsopoulos, Konstantinos, and Michael N Vrahatis. "Multi-Objective Particle Swarm Optimization Approaches Interval Methods for Resolving Neural Network Issues View Project Metaheurestic Optimization in Machine Learning View Project." doi:10.13140/2.1.5189.4721.

Rao, R.V., Savsani, V.J. and Vakharia, D.P., 2011. Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems. Computer-aided design, 43(3), pp.303-315.

Reddy, M. Janga, and D. Nagesh Kumar. 2006. “Multi-Objective Optimization Using Evolutionary Algorithms.” *Water Resources Management* 20 (6): 861–878.

Simões, Ana Correia, Ana Pinto, Joana Santos, Sofia Pinheiro, and David Romero. 2022. “Designing Human-Robot Collaboration (HRC) Workspaces in Industrial Settings: A Systemic Literature Review.” *Journal of Manufacturing Systems* 62 (January). Elsevier B.V.: 28–43. doi:10.1016/j.jmsy.2021.11.007.

Stecke, Kathryn E., and Mahdi Mokhtarzadeh. 2022. “Balancing Collaborative Human–Robot Assembly Lines to Optimise Cycle Time and Ergonomic Risk.” *International Journal of Production Research* 60 (1). Taylor and Francis Ltd.: 25–47. doi:10.1080/00207543.2021.1989077.

Sun, Ying, and Yuelin Gao. 2019. “A Multi-Objective Particle Swarm Optimization Algorithm Based on Gaussian Mutation and an Improved Learning Strategy.” *Mathematics* 7 (2). MDPI AG. doi:10.3390/math7020148.

Taheri, Ahmad, Keyvan RahimiZadeh, and Ravipudi Venkata Rao. 2021. “An Efficient Balanced Teaching-Learning-Based Optimization Algorithm with Individual Restarting Strategy for Solving Global Optimization Problems.” *Information Sciences* 576 (October). Elsevier Inc.: 68–104. doi:10.1016/j.ins.2021.06.064.

Tang, Qiuhua, Zixiang Li, Li Ping Zhang, and Chaoyong Zhang. 2017. “Balancing Stochastic Two-Sided Assembly Line with Multiple Constraints Using Hybrid Teaching-Learning-Based Optimization Algorithm.” *Computers and Operations Research* 82 (June). Elsevier Ltd: 102–113. doi:10.1016/j.cor.2017.01.015.

Tang, Q., Meng, K., Cheng, L. and Zhang, Z., 2022. An improved multi-objective multifactorial evolutionary algorithm for assembly line balancing problem considering regular production and preventive maintenance scenarios. Swarm and Evolutionary Computation, 68, p.101021.

Vieira, Miguel, Samuel Moniz, Bruno S. Gonçalves, Tânia Pinto-Varela, Ana Paula Barbosa-Póvoa, and Pedro Neto. 2021. “A Two-Level Optimisation-Simulation Method for Production Planning and Scheduling: The Industrial Case of a Human–Robot Collaborative Assembly Line.” *International Journal of Production Research*. Taylor and Francis Ltd. doi:10.1080/00207543.2021.1906461.

Weckenborg, Christian, Karsten Kieckhäfer, Christoph Müller, Martin Grunewald, and Thomas S. Spengler. 2020. “Balancing of Assembly Lines with Collaborative Robots.” *Business Research* 13 (1). Springer: 93–132. doi:10.1007/s40685-019-0101-y.

Weckenborg, Christian, and Thomas S. Spengler. 2019. “Assembly Line Balancing with Collaborative Robots under Consideration of Ergonomics: A Cost-Oriented Approach.” In *IFAC-PapersOnLine*, 52:1860–1865. Elsevier B.V. doi:10.1016/j.ifacol.2019.11.473.

Zhang, Ya jun, Ningjian Huang, Rober G. Radwin, Zheng Wang, and Jingshan Li. 2022. “Flow Time in a Human-Robot Collaborative Assembly Process: Performance Evaluation, System Properties, and a Case Study.” *IISE Transactions* 54 (3). Taylor and Francis Ltd.: 238–250. doi:10.1080/24725854.2021.1907489.

Zhang, Zikai, Qiu Hua Tang, Rubén Ruiz, and Liping Zhang. 2020. “Ergonomic Risk and Cycle Time Minimization for the U-Shaped Worker Assignment Assembly Line Balancing Problem: A Multi-Objective Approach.” *Computers and Operations Research* 118. Elsevier Ltd: 104905. doi:10.1016/j.cor.2020.104905.

Declaration of interests

☒The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: