PRODUCTION ENGINEERING ARCHIVES 2026, 32(2), 335-341

PRODUCTION ENGINEERING ARCHIVES

ISSN 2353-5156 (print)
ISSN 2353-7779 (online)

Exist since 4th quarter 2013
Available online at https://pea-journal.eu



An Efficient Tree Search Algorithm for Solving Robotic Assembly Line Balancing Problems

Hedi Mhalla[1*], Nilüfer Pekin Alakoç[2]

[1] College of Engineering and Technology, American University of the Middle East, Kuwait; hedi.mhalla@aum.edu.kw
[2] Bursa Uludağ University, Bursa, Türkiye; Niluferalakoc@uludag.edu.tr
*Correspondence: hedi.mhalla@aum.edu.kw

Article history
Received 06.01.2026
Accepted 17.04.2026
Available online 11.05.2026

Keywords
Robotic lines,
Assembly line balancing,
Heuristic algorithm,
Equipment requirements,
Integer programming.

Abstract
Traditional assembly line balancing problems (ALBP) aim to assign tasks to stations to meet specific production targets, but in practice, the problems can be extremely complex because of the additional factors, such as robot or automated equipment alternatives. The robotic assembly line problem is a broad and challenging variant of traditional lines. Designing and balancing robotic assembly lines are crucial in manufacturing to optimize productivity, efficiency, and flexibility. In this study, we propose a heuristic algorithm that provides practical and effective solutions for robotic assembly line balancing problems (RALB). We aim to assign robot and task combinations to workstations simultaneously with the objective of minimizing the system cost, including the cost of installing new robots and opening new workstations. We evaluate the algorithm's performance on a large set of random problem instances by using statistical methods. We conclude that feasible and good solutions can be found easily, even for large-scale problems, in short processing times.

DOI: 10.30657/pea.2026.32.26

1. Introduction

Over time, production systems have become more complex with the increasing variety of products and manufacturing alternatives. This complexity is caused by the constantly changing customer demands for various and customized products and manufacturers' desire to design production systems to increase productivity and produce a wide range of products. Therefore, manufacturers' efficient use of resources is the main focus when managing these complex assembly lines. The importance of balancing the assembly lines and optimizing the usage appears at this point.

The ALBP is essentially about assigning tasks to workstations to find effective solutions for a specific objective, without violating the line's constraints. However, each assembly line has its unique structure and requirements, leading to various versions of the basic problem in the literature. These variants consider different objectives and characteristics such as product, layout, workflow, and task time. As a result, a multitude of problems and solution methods have been explored in the literature (Battaïa and Dolgui, 2013).

In this study, we mention the importance of realistic assembly lines by including robot allocation and aim to expand the literature with a new perspective. Our main problem is to assign tasks simultaneously with the robots to workstations to minimize the total cost. Our RALBP is challenging due to the difficulty caused by cost and robot alternatives. Moreover, most ALBP are known as NP-hard (Álvarez-Miranda and Pereira 2019). So, we develop a new heuristic algorithm to address this issue and find approximate solutions for large-scale problems in a reasonable amount of time. A closely related study is presented by Pekin and Mhalla (2025). The main distinctions between the two approaches concerns the overall algorithmic structure, procedural steps, and framework. In their work, the authors introduced two strategies for initiating the solution: a conventional greedy approach and a balanced greedy approach. These initial solutions were subsequently refined using a standard neighbourhood search procedure. Finally, an assignment procedure based on a reduced and simple beam search is employed to assign any remaining tasks, if any.

In contrast, the proposed method utilizes a more integrated and cohesive framework. The construction phase is entirely driven by a consistent beam search procedure that incremen-

© 2026 Author(s). This is an open access article licensed under the Creative Commons Attribution (CC BY) License (https://creativecommons.org/licenses/by/ 4.0/).

tally generates high-quality partial solutions. This phase is followed by a neighbourhood search improvement stage to further enhance solution quality. Additionally, the algorithm incorporates advanced selection and evaluation criteria during construction, which ensures greater diversification and facilitates broader exploration of the solution space. Consequently, the proposed approach achieves a more effective balance between intensification and diversification compared to the sequential strategy employed by Pekin and Mhalla (2025).

2. Literature review

One of the branches of simple assembly line balancing problems (SALBP) includes the robot concept. RALBP is the assignment of tasks to workstations along with robots. The solution to this problem is an attempt to find the optimum selection. Naturally, task times depend on the robot alternatives and allocation costs, which are the main restrictions in RALBP. This problem is not new in the literature, it has received attention since 1979. Graves and Whitney (1979), Graves and Lamar (1983), Pinto et al. (1983), Graves and Holmes Redfield (1988), and Rubinotitz and Bukchin (1993) are the former researchers who addressed equipment/robot decisions. They modified the SALBP by adding a selection and assignment of processing alternatives. Graves and Whitney (1979) suggested a branch and bound algorithm, and Graves and Lamar (1983) discussed an approximate solution procedure for the extended problem. Graves and Whitney (1983) and also Pinto et al. (1983) aimed to minimize system costs. Different from the others, Graves and Holmes Redfield (1988) modelled a problem including several products. Later, Bukchin and Tzur (2000) developed an exact and heuristic method for assigning one equipment to each station along with tasks. Rekiek et al. (2002) discussed the selection of operating modes, such as manual, robotic, and automated by a hybrid assembly line design. Levitin et al. (2006), Gao et al. (2009), Nilakantan and Ponnambalam (2012), Borba et al. (2018), and Lahrichi et al. (2023) focused on minimizing the cycle time, aiming to improve robotic line efficiency under different modelling assumptions. The proper selection of robots/equipment has great importance for effective assembly lines and system costs, hence some researchers mainly considered the problem from an economic point of view and developed algorithms to minimize the equipment or robot costs such as Barutçuuoğlu and Azizoğlu (2011), Li et al. (2011), Battaïa et al. (2013), Pereira et al. (2018), and Alakoç Pekin and Mhalla (2025). Ege et al. (2009), Bukchin and Rubinovitz (2002), and Battaïa et al. (2014) extended the basic problem to resource decisions for parallel stations. Besides straight line layouts, U-shaped lines with equipment alternatives were also studied (Ogan and Azizoğlu 2015; Kara et al. 2012; Jayaswal and Agarwal 2014; Chiang et al. 2007; and Nilakantan and Ponnambalam, 2016). Furthermore, most researchers developed models with a larger perspective and considered multiple objectives, such as Barutçuoğlu and Azizoğlu (2011), Pekin and Azizoğlu (2008), Yoosefelahi et al. (2012), Chehade et al. (2012), Triki et al. (2017), Mura and Dini (2016), Oesterle et al. (2016), Çil et al. (2016), and Zhou and Wu (2020).

3. Problem statement

We extend the RALBP by including the flexibility of allocation of multiple robots to one station. We aim to assign task-robot pairs to workstations to minimize total cost which includes robot allocation and new workstation opening costs.

Assumptions:

- The line is straight.
- Tasks must be performed by the precedence relations.
- Every task is assigned to a unique robot and produced in a unique station.
- Processing times are deterministic and vary according to the robot.
- Cycle time is constant.
- Stations are identical, each station can perform any task.
- Robots have variable costs which include setup, purchasing, maintenance, and consumption costs.
- Robot allocation costs are constant for each station.
- The number of robots assigned to any workstation is not limited. More than one robot can be assigned to a workstation. However, each robot type can be assigned to a workstation at most once.
- All other times and factors are negligible: material transportation, loading, unloading, and robot setup times and required floor space … etc.

Notations:

- n*:* Number of tasks
- r*:* Number of robots
- K: Maximum possible number of workstations
- C: The cycle time
- t_{ij}: Processing time of task *i* by robot *j*.
- *SC*: Station opening cost
- RC_j: Cost of robot *j*.
- $x_{ijk} = \begin{cases} 1 & \text{If task } i \text{ is assigned to station } k \text{ by robot } j. \\ 0 & \text{otherwise} \end{cases}$
- $y_{jk} = \begin{cases} 1 & \text{If robot } j \text{ is assigned to workstation } k. \\ 0 & \text{otherwise} \end{cases}$
- $z_k = \begin{cases} 1 & \text{If workstation } k \text{ is used.} \\ 0 & \text{otherwise} \end{cases}$
- *S*: Number of workstations
- $i = 1, \dots, \mathrm{n}$
- $j = 1, \dots, \mathrm{r}$
- $k = 1, \dots, \mathrm{K}$

Linear Mathematical Model:

$$\text{Min } \sum_{k=1}^{K} \sum_{j=1}^{r} \left(RC_j * y_{jk} \right) + S * SC \quad (1)$$

Subject to

$$\sum_{k=1}^{K} \sum_{j=1}^{\mathrm{r}} x_{ijk} = 1 \qquad \forall i \quad (2)$$

$$\sum_{i=1}^{\mathrm{n}} \sum_{j=1}^{\mathrm{r}} t_{ij}\, x_{ijk} \leq C \qquad \forall k \quad (3)$$

$$\sum_{k=1}^{K} z_k = S \quad (4)$$

$$\sum_{j=1}^{\mathrm{r}} \sum_{k=1}^{K} k x_{ajk} \leq \sum_{j=1}^{\mathrm{r}} \sum_{k=1}^{K} k x_{bjk} \qquad \forall (a, b), \text{ task a precedes task b} \quad (5)$$

$$\sum_{j=1}^{r} y_{jk} \leq rz_k \qquad \forall k \qquad (6)$$

$$x_{ijk} \leq y_{jk} \qquad \forall i,j,k \qquad (7)$$

$$1 \leq S \leq \mathrm{K} \qquad (8)$$

$$x_{ijk} \in \{0,1\} \qquad \forall i,j,k \qquad (9)$$

$$y_{jk} \in \{0,1\} \qquad \forall j,k \qquad (10)$$

$$z_k \in \{0,1\} \qquad \forall k \qquad (11)$$

$$S \in \mathbb{Z}^+ \qquad (12)$$

Equation 1 describes the objective function. Equation 2 gives the constraints that provide the unicity of tasks' assignments to workstations. Equation 3 ensures that the workstation loads do not exceed the capacity. Constraints given in Equation 4 state that processing all tasks needs at least S workstations. Equation 5 guarantees precedence relations for all the tasks. Constraints given in Equation 7 ensure that tasks and required robots to process the tasks are assigned together to the same workstation. Equation 8 bounds S. Equations 9, 10, and 11 define binary variables and Equation 12 specifies that *S* is a positive integer.

4. The algorithm and the solution procedure

The algorithm is composed of four main steps and calculates the minimum number of stations and robots with the minimum costs that are needed to schedule all tasks. We proceed with a pre-treatment of the data, where all the tasks are sorted in a specific order to simplify the algorithm and decrease the number of elementary operations performed by the algorithm. To achieve our sorting, we apply a two-dimensional sorting. The initial one is vertical sorting, which concerns the tasks and targets to provide an indexation based on the precedence constraints. Then, the subsequent sorting is a horizontal one, which is applied to each task to give an increasing order to the robots based on the time needed to achieve the tasks. In the next step, the algorithm constructs an initial solution formed on the best robot by the predefined horizontal sorting. Finally, we apply a local search procedure to improve the solution.

Lower bound on the number of stations:

To address our problem, we need to have a practical estimation of the minimum number of stations required to complete all the tasks. Our first step is to calculate a time-based greedy solution that finds the minimum time needed for each task while disregarding other constraints. Essentially, we obtain the total of minimum task times for all tasks which is denoted as $Min(Tt)$ and calculated by $Min(Tt) = \sum_{i=1}^{n} min_{j=1,\dots,r}(t_{ij})$. Then, we determine the minimum number of stations required, denoted by $Min(S)$ and defined as given in Equation 9.

$$Min(S) = \begin{cases} \frac{Min(Tt)}{C}, & \text{if } (Min(Tt)/C) \text{ is an integer} \\ \left\lceil \frac{Min(Tt)}{C} \right\rceil + 1, & \text{otherwise} \end{cases} \qquad (9)$$

Similarly, we calculate *K*, the maximum number of workstations needed, by calculating a time-based greedy solution that finds the maximum time needed for each task while disregarding other constraints.

Vertical sorting:

This procedure helps to minimize the amount of data exploration required in the dataset matrix. The initial order of tasks is maintained, and tasks are assigned indexes based on their precedence constraints such that tasks a and b have respective indexes i_1 and i_2 where $i_1 < i_2$, if task a needs to be accomplished before task b. This ensures that tasks are completed in the correct order without the need for extensive exploration of the data.

Horizontal sorting:

As we have already discussed in this document, it is possible to perform any task with different robots. So, for every task, we sort the robots in increasing order of one criterion. We refer to this sorting as horizontal sorting. Different ways of horizontal sorting are possible, such as increasing order of execution time or robot cost for each task. In this step, we propose to use the one that is based on the task times. For each task i, the robots are sorted and indexed in such a way that robots j_1, ..., j_r, satisfy: $t_{ij_1} \leq \cdots \leq t_{ij_r}$.

The Beam Search Algorithm:

After the horizontal sorting of tasks, we build a greedy solution for each one of the stations where the lower limit of the number of stations is as given in Equation 9. We identify the robot to allocate to every station based on each robot's repetitions (frequency) as first ranked in our horizontal sorting. For each task, we determine a predefined fixed number of the best options, ω and name it as the algorithm's width. To allocate tasks to stations efficiently, we follow a specific process. First, we assign one robot to a station based on the frequency of the first item in the horizontal sorting list. We then select tasks for each station and robot based on their task time. We also consider the precedence constraints by the vertical sorting. A station with a lower load receives the next sorted task. If we cannot assign a task to one of the stations where the currently allocated robot(s) is (are) not the first of the horizontal sorting, then two branches are generated in the solution. In the first branch, we add a new robot if it is possible to perform a task with the lowest index robot, and we consider the station with the highest residual time and, if needed, open a new station with the same robot. The second branch corresponds to the allocation of the task to the station with the robot having the following lowest index based on the horizontal sorting and which has a remaining load. Then, the two branches are developed as sons derived from this node.

To decide where to assign a task when there is more than one option, we use a priority rule; selecting the station with the highest residual time, which also helps to balance the stations' loads.

While generating solution branches, other priority rules, such as selecting the station with the lowest residual time, can be considered, but as our preliminary tests confirmed, the balancing approach remains the most appropriate due to the nature of the problem.

It is evident that as the number of branches increases, the algorithm's complexity also increases, leading to longer execution times. As we mentioned earlier, we performed a series of preliminary tests on various instances. The primary aim of these tests was to fine-tune, optimize, and evaluate the performance of the algorithm.

One crucial parameter is the algorithm's width, ω, derived from each task. We keep progressing with the best ω partial solutions at each assignment level. This process ensures that tasks are allocated efficiently and effectively. It also allows us to identify the branches that provide the best solutions. Through this process, we also ensure a diversification of the search process.

The local search procedure:

Finally, we apply an improvement procedure by exchanging tasks or their allocations (station and/or robots). We suggest this procedure to simplify the search and reduce the time needed to improve the current set of solutions as much as possible. The heuristic named adaptive 2-Opt was initially applied to resolve transportation problems (Lin, 1965), and it has been shown to give good results. The 2-Opt was also applied to other complex combinatorial optimization problems and again provided valuable improvements. The 2-Opt is a process that allows the exchange of the allocation of two tasks within the current solution. Here, we exchange one task from one of the stations with another one from a different station. The objective of the adaptive 2-Opt is to group the tasks using the same robots, which, if applicable, reduces the cost. The process is repeated for the tasks or sequences of tasks in stations with multiple robots. The procedure where the number of iterations is not predefined is very fast, not time-consuming, and provides strong efficiency.

The Resolution Algorithm:

This section presents a summary of the resolution algorithm, where the main steps are listed as follows. All through the algorithm, we use the following notations:

- M is the set of indexes of the not assigned tasks.
- N is the set of elected partial solutions.
- z is a feasible solution.
- $C(z)$ is the total cost of the solution z.

Algorithm 1 summarizes the important steps of the resolution algorithm.

Algorithm 1: The resolution Algorithm

Input:

- Set $N = \emptyset$, and $M = \{1, \dots, m\}$;
- $Min(S)$; K
- Set of n task types
- Processing times t_{ij}

Output: (S) and a schedule (chronological assignment of tasks to stations and robots)

Steps:

- Horizontal and vertical sorting. We considered a fixed proportion of tasks with precedence constraints, set at
- Calculate the greedy solution based on the minimum time allocated to each task.
- Calculate the minimum number of stations needed $Min(S)$.
- Calculate K, the maximum number of stations.
- Attribute one robot per station based on the frequencies of the lowest-cost robots for each task.
- Let $i = 1$. (We start with the available task and with the lowest index)
- Iterative process: $While\ (i \leq m)$
  - Allocate the task i to one of the existing stations while considering all the available robots. (If more than one station is available, consider the lowest load station. If task allocation is not possible to the current stations, a new station is opened with the corresponding robot considered in the solution.)
  - Evaluate the partial solutions.
  - Rank the current nodes in the non-decreasing order.
  - Update N. (N is the set of the current best ω partial solutions).
  - $i = i + 1$.
- Apply the 2-Opt procedure to reduce the number of robots per station, if possible.

In summary, the algorithm starts with the data's preprocessing, which mainly consists of the sorting step. This is followed by the greedy approach to identify the minimum time needed to achieve all tasks and the minimum number of stations. We initialize the station and the associated robots. Then, we apply the iterative steps of the beam search procedure. Finally, we apply the local search procedure to improve the current solution. The algorithm exits with a solution for the problem's instance.

5. Experiments, results and discussion

In this section, we discuss the behavior and efficiency of the algorithm. The algorithm was coded in C++ using the AUM-Phenix HPC facility (10 Dell PowerEdge R730 servers, Intel Xeon E5-2698 v3 2.3GHz). We randomly generated instances with task numbers n =100, 250, 500, 750, and 1000, with possible robot numbers r =6, 8, and 10. We generated twenty instances for each pair of values of n and r to analyze the algorithm's performance regarding the capability to solve the instances and the solution efficiency. We considered a fixed proportion of tasks with precedence constraints, set at 25%. That is, in each generated instance, 25% of the tasks were randomly selected and assigned random precedence relationships, ensuring that the precedence graphs reflect a moderate level of task dependency. The regular processing times are randomly generated following a uniform distribution within the interval [5, 25], and regular robot costs follow a uniform distribution within [40, 100]. The station-opening costs are assumed to be the same across all stations and were determined based on the robot cost distribution. Specifically, the median of the randomly generated robot costs of the instance was multiplied by a uniform random variable from the interval [4, 8]. Finally, the stations' cycle times, which are also identical for all stations, were generated randomly. Similar to the station

opening cost generation, the median of tasks' minimum processing times was multiplied by a uniformly distributed random variable from the interval [10, 15].

We compared the algorithm's final solution with an upper and lower bound denoted by U and L, respectively. The upper one is a solution calculated by a greedy algorithm that creates stations based on the need and allocates the robot with the cheapest cost. It corresponds to a simplified version of the main algorithm tested here, where no improvements are applied. This means we are focusing more on the feasibility of the solution than its quality. We also implemented an aggressive lower bound based on the minimum number of stations needed to achieve the tasks and allocate the robot with the lowest cost to each one of the stations. We relaxed all the other constraints related to tasks and their execution. The solution is generally not feasible, but provides an excellent indicator to measure the algorithm's performance. We define the performance of a solution, z, with an objective function's value $C(z)$, as:

$$P(\mathrm{z}) = \left(1 - \frac{(C(z)-L)}{(U-L)}\right) * 100 \qquad (10)$$

We clarify to the readers that P(z) is not intended solely as a standalone absolute measure of solution quality, but rather as a normalized indicator that facilitates comparison with prior studies and across instance classes.

Table 1 presents the summary of the experimental results. Column 1 gives the size of the problem instance; column 2 shows the number of robots considered; column 3 shows the average performance of the problem instances generated for each category. Finally, columns 4 and 5 display $P(z)$'s minimum and maximum, respectively.

Table 1. Summary of the beam search algorithm

n	***r***	$Av(P(z))$	$Min(P(z))$	$Max(P(z))$
100	*6*	29.93	19.08	40.15
	8	27.32	23.98	37.64
	10	23.71	21.45	33.04
250	*6*	14.42	09.63	29.87
	8	37.10	17.51	43.19
	10	23.17	19.49	31.73
500	*6*	18.21	15.29	30.49
	8	17.09	12.36	35.05
	10	24.95	15.81	39.57
750	*6*	11.87	03.37	36.59
	8	28.31	21.81	42.63
	10	32.79	23.45	45.02
1000	*6*	13.11	09.97	30.73
	8	12.56	06.94	32.14
	10	14.61	08.58	35.71

Table 1 shows the ability of the algorithm to improve the initial solutions for each combination of the number of robots and tasks. These results are computed at the unique level of the exploration width ω =3. We discuss the effect of the parameter ω at the end of this section. As expected, the number of tasks affects the algorithm's performance; it is lower when the *n* is 1000, slightly better for 100, and for the other task times there are no significant differences. This is due to the increasing feasible solutions in the presence of more tasks and robots. The same pattern is observed for the minimum and maximum numbers. Since the statistics don't show significant differences, we cannot say that the number of robots is a factor for the performance.

We also summarize and evaluate the overall performance of the algorithm by the main descriptive statistics (Table 2).

Table 2. Overall summary statistics

Ave	*Median*	*Min*	*Max*	*Std. dev.*	***95% CI***
21.94%	30.59%	3.37%	45.02%	12.39%	[20.58, 23.30]

The results demonstrate that $P(z)$ varies from 3.37% to 45.02%; The overall minimum is observed for 750 tasks and 6 robot alternatives, and the overall maximum is 45.02% for 750 tasks and 10 robots. Table 2 gives an average value of 21.94%, a median value of around 30.59%, and an overall standard deviation of 12.39%, meaning a relatively high variability. The 95% confidence interval provides a small range for the overall mean. Moreover, the data indicate a left-skewed distribution and 50% of the data falls on the right side of 30.59%. The variability is mainly induced by the poor quality of the lower bound, which we calculate by an aggressive relaxation of several constraints related to the task execution, such as the precedence constraints. Globally, the algorithm's performance remains consistent and the improvement procedure on the initial solution gives relatively good results.

In addition to descriptive statistics, we conducted some tests to evaluate the performance of the algorithm to the changes in *n* and *r*. The effects of *n*, *r*, and the interaction between *n* and *r* on the performance were tested by a balanced two-way ANOVA. The results given in Table 3 can be summarized as follows: The main effects of *n* and *r* are statistically significant at a significance level of 0.05. Similarly, there exists significant interaction effect between *n* and *r* (with a small p-value less than 0.005).

Table 3. Two-way ANOVA table of the algorithm

Source	**DF**	**SS**	**MS**	**F**	**P-value**
n	4	6948.900	1737.225	544.263	< 1.0E-15
r	2	2970.672	1485.336	465.347	< 1.0E-15
n x r	8	8274.190	1034.274	324.032	< 1.0E-15
Error	285	909.688	3.192		
Total	299	19103.451			

We also examined the assumptions of two-way ANOVA. Independence of the instances was ensured by the design of the experiments. The normality assumption was assessed using the Shapiro–Wilk test, which yielded a p-value of 0.4823, indicating that the residuals are approximately normally distributed. Although the experimental design is fully balanced with 20 replications, under which the two-way ANOVA is

known to be robust to moderate violations of homoscedasticity, the homogeneity of variances was tested using Levene's test. As expected, a violation was not detected. The resulting large p-value (0.6241) indicates that the variances are approximately equal across all combinations of *n* and *r*.

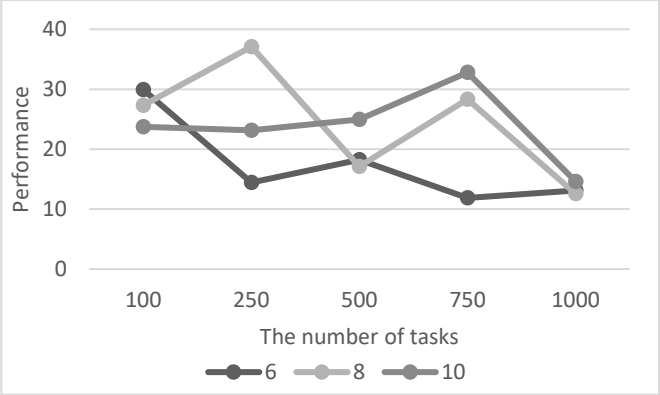


Fig. 1. Interaction plot of the performance

In addition, we used the interaction plot to examine the joint effect of *n* and *r* on the average performance measure. Figure 1 illustrates non-parallel and crossing lines, indicating the presence of a strong interaction effect between the two factors. The effect of r on performance depends on the problem size. For small (*n*=100) and large (*n*=1000) problem size differences among the *r* are relatively small. Except for small size problems, higher values of r (*r*=8, 10) lead to significantly higher performance outcomes, particularly for *n*=250 and *n*=750. Moreover, as n increases, the performance measure decreases for all r values.

Due to the significant interaction between *n* and *r*, we examined simple effects and performed post-hoc comparisons by using Tukey's HSD test. Based on the HSD=1.93 at the 5% significance level. The results indicate that, for almost all levels of n, the performance differs significantly across *r*. All *r* values yield significantly different performance for *n* =100, 250, and 750, whereas for *n*=500 and 1000, some *r* values show no significant differences. These are for *n*=500 *r*=6 vs *r*=8 and for *n*=1000 *r*=6 vs *r*=8, and *r*=6 vs *r*=10.

These findings and the non-parallel pattern of the interaction plot confirm the significant interaction effect observed in the ANOVA.

It is essential to mention that the results of Table 1 and Table 2 are for the unique level of the algorithm's width. It is evident that by the construction of the algorithm, the solution's quality increases when ω increases because we explore more options and develop more branches. We used pair-wise tests applied to the set of problem instances and determined that a width of 3 is a good choice for the algorithm. The statistical analyses revealed that the algorithm's performance when ω=1 and ω=2 is relatively weak regarding solution improvements, which means that the final solution is very close to the upper bound, inducing a limited capacity of improvements with a $P(z)$ that varies from 1% to 13%. The paired-wise test for the $P(z)$ values for ω =2 and ω =3, at the significance level of 0.05, shows a clear dominance for ω =3 with 10% or more. For ω =4 to 6, and ω =3, for the same significance level, the pair-wise test result shows no significant improvements greater than 2%. So, we concluded that the width value of ω =3 provides a good balance for the proposed algorithm.

6. Conclusion

The ALBP is a well-studied topic in manufacturing research, with many studies examining various methods for optimizing assembly lines. An important part of the studies includes a combination of various extensions and alternative resources. Considering these variants can significantly impact complex production systems' overall cost and efficiency.

This paper presents a novel heuristic algorithm designed specifically to address the RALBP. Our approach is to minimize the total robot allocation and new station opening costs. We begin by establishing a lower bound on the number of stations, a unique starting point for the method. Then we strategically sort the tasks and robots for each task to minimize the time and solution alternatives. The robots and tasks are assigned to the stations using the beam search algorithm, and we conclude by applying the adaptive 2-Opt procedure to further enhance the solution.

To evaluate our algorithm's performance, we generated a large set of random problem instances for different combinations of the number of robots, tasks, and the best solutions. We then conducted comprehensive statistical analyses to assess the algorithm's performance rigorously. As we expected, increasing the number of tasks and robot alternatives extends the dimensions and size of the problem considerably. The results of our analyses show that the algorithm is satisfactory enough to reduce the complexity of the problem and able to solve large problems in a reasonable time.

For future work, we recommend extending the problem by adding practical requirements for the lines, such as restrictions on the allocation of sets of robots on the same workstation. Moreover, minimizing the cycle time can also be considered for a multi-objective problem. Other improvement procedures can also be included to increase the quality of the solutions.

Reference

Álvarez-Miranda, E., Pereira, J., 2019. On the complexity of assembly line balancing problems. Computer and Operations Research, 108, 182-186. DOI: 10.1016/j.cor.2019.04.005

Barutçuoğlu, S., Azizoğlu, M., 2011. Flexible assembly line design problem with fixed number of workstations. International Journal of Production Research, 49, 3691-714. DOI: 10.1080/00207543.2010.492410

Battaïa, O., Dolgui, A., 2013. A taxonomy of line balancing problems and their solution approaches. International Journal of Production Economics, 142, 259-277. DOI: 10.1016/j.ijpe.2012.10.020

Battaïa, O., Dolgui, A., Guschinsky, N., Levin, G., 2014. Integrated configurable equipment selection and line balancing for mass production with serial–parallel machining systems. Engineering Optimization, 46(10), 1369-1388. DOI: 10.1080/0305215X.2013.841904

Battaïa, O., Gurevsky, E., Makssoud, F., Dolgui, A., 2013. Equipment location in machining transfer lines with multi-spindle heads. Journal of Mathematical Modelling and Algorithms in Operations Research, 12. DOI: 10.1007/s10852-012-9196-2

Borba, L., Ritt, M., Miralles, C., 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

Bukchin, J., Rubinovitz, J., 2002. A weighted approach for the assembly line design with station paralleling and equipment selection, IIE Transactions, 35(1), 513-524. DOI: 10.1080/07408170304429

Bukchin, J., Tzur, M., 2000. Design of flexible assembly line to minimize equipment cost, IIE Transactions, 32, 585-598. DOI: 10.1023/A:1007646714909

Chehade, H., Dolgui, A., Dugardin, F., Makdessian, L., Yalaoui, F., 2012. Multi-objective approach for production line equipment selection. Management and Production Engineering Review, 3(1), 417. DOI: 10.2478/v10270-012-0001-5

Chiang, W., Kouvelis, P., Urban, T., 2007. Line balancing in a just-in-time production environment: balancing multiple U-lines. IIE Transactions 39(4), 347-359. DOI: 10.1080/07408170600838340

Çil, Z.A., Mete, S., Ağpak, K., 2016. A goal programming approach for robotic assembly line balancing problem. IFAC Papers On Line 49(12), 938-942. DOI: 10.1016/j.ifacol.2016.07.896

Ege, Y., Azizoğlu, M., Özdemirel, N., 2009. Assembly line balancing with station paralleling, Computers & Industrial Engineering, 57, 1218-1225. DOI: 10.1016/j.cie.2009.05.014

Gao, J., Sun, L., Wang, L., Gen, M., 2009. An efficient approach for type II robotic assembly line balancing problems. Computers & Industrial Engineering, 56(3), 1065-1080. DOI: 10.1016/j.cie.2008.09.027

Graves, S.C., Holmes Redfield, C., 1988. Equipment selection and task assignment for multiproduct assembly system design. International Journal of Flexible Manufacturing Systems, 1, 31-50. DOI: 10.1007/BF00713158

Graves, S.C., Lamar, B.W., 1983. An integer programming procedure for assembly system design problems. Operations Research, 31, 522-545. DOI: 10.1287/opre.31.3.522

Graves, S.C., Whitney, D.E., 1979. A mathematical programming procedure for the equipment selection and system evaluation in programmable assembly. Proceedings of the Eighteenth IEEE Conference on Decision and Control, Ft Lauderdale, FL., 531-536. DOI: 10.1109/CDC.1979.270236

Jayaswal, S., Agarwal, P., 2014. Balancing U-shaped assembly lines with resource dependent task times: a simulated annealing approach. Journal of Manufacturing Systems, 33, 522-34. DOI: 10.1016/j.jmsy.2014.05.002

Kara, Y., Özgüven, C., Yalçın, N., Atasagün, Y., 2012. Balancing straight and U-shaped assembly lines with resource dependent task times. International Journal of Production Research, 49, 6387-405. DOI: 10.1080/00207543.2010.535039

Levitin, G., Rubinovitz, J., Shnits, B., 2006. A genetic algorithm for robotic assembly line balancing. European Journal of Operational Research, 168(3), 811-825. DOI: 10.1016/j.ejor.2004.07.030

Lin, S., 1965. Computer solutions of the traveling salesman problem. In The Bell System Technical Journal, 44(10):2245-2269, DOI: 10.1002/j.1538-7305.1965.tb04146.xa

Li, S., Wang, H., Hu, S., Lin, Y., Abell, J., 2011. Automatic generation of assembly system configuration with equipment selection for automotive battery manufacturing. Journal of Manufacturing Systems, 30(4), 188-195. DOI: 10.1016/j.jmsy.2011.07.009

Mura, M.D., Dini, G., 2016. Worker skills and equipment optimization in assembly line balancing by a genetic approach. Procedia CIRP, 44, 102-107. DOI: 10.1016/j.procir.2016.02.033

Nilakantan, J.M., Ponnambalam, S.G., 2012. An efficient PSO for type II robotic assembly line balancing problem. Proceedings of the IEEE international conference on automation science and engineering (CASE), Seoul, Korea (South), 600-605. DOI: 10.1109/CoASE.2012.6386398

Nilakantan, J.M., Ponnambalam, S.G., 2016. Robotic U-shaped assembly line balancing using particle swarm optimization. Engineering, 48(2), 231-252. DOI: 10.1080/0305215X.2014.998664

Oesterle, J., Bauernhansl, T., Amodeo L., 2016. Hybrid multi-objective optimization method for solving simultaneously the line balancing, equipment and buffer sizing problems for hybrid assembly systems. Procedia CIRP, 57, 416-421. DOI: 10.1016/j.procir.2016.11.072

Ogan, D., Azizoğlu, M., 2015. A branch and bound method for the line balancing problem in U-shaped assembly lines with equipment requirements. Journal of Manufacturing Systems, 36. DOI: 10.1016/j.jmsy.2015.02.007

Pekin, N., Azizoğlu, M., 2008. Bi criteria flexible assembly line design problem with equipment decisions. International Journal of Production Research, 46(22), 6323-6343. DOI: 10.1080/00207540701441988

Pekin Alakoç, N., Mhalla, H., 2025. A heuristic approach for solving robotic assembly line balancing problems. Engineering Technology & Applied Science Research, 15(2). DOI: 10.48084/etasr.9845

Pereira, J., Ritt, M., Vásquez, O.C., 2018. A memetic algorithm for the cost-oriented robotic assembly line balancing problem. Computers & Operations Research, 99, 249-261. DOI: 10.1016/j.cor.2018.07.001

Pinto, P.A., Dannenbring, D.G., Khumawala, B.M., 1983. Assembly line balancing with processing alternatives: an application. Management Science, 29, 817-830. DOI: 10.1287/mnsc.29.7.817

Rekiek, B., Dolgui, A., Delchambre, A., Bratcu, A., 2002. State of art of optimization methods for assembly line design. Annual Reviews in Control, 26, 163-174. DOI: 10.1016/S1367-5788(02)00027-5

Rubinovitz, J., Bukchin, J., 1993. RALB - a heuristic algorithm for design and balancing of robotic assembly lines. Annals of the CIRP 42, 497-500. DOI: 10.1016/S0007-8506(07)62494-9

Triki, H., Mellouli, A., Masmoudi, F., 2017. A multi-objective genetic algorithm for assembly line resource assignment and balancing problem of type 2 (ALRABP-2). Journal of Intelligent Manufacturing, 28(2), 371-385. DOI: 10.1007/s10845-014-0984-6

Yoosefelahi, A., Aminnayeri, M., Mosadegh, H., Davari Ardakahi, H., 2012. Type II robotic assembly line balancing problem: An evolution strategies algorithm for a multi-objective model. Journal of Manufacturing Systems, 31(2), 139-151. DOI: 10.1016/j.jmsy.2011.10.002

Zhou, B., Wu, Q., 2020. Decomposition-based bi-objective optimization for sustainable robotic assembly line balancing problems. Journal of Manufacturing Systems, 55, 30-43. DOI: 10.1016/j.jmsy.2020.02.005