Stochastic Local Search for Solving Chance-Constrained Multi-Manned U-shaped Assembly Line Balancing Problem with Time and Space Constraints

Mohammad Zakaraia[1], Hegazy Zaher[2], Naglaa Ragaa[3]

[1]PhD Candidate in Operations Research, faculty of graduate studies for statistical research, Cairo University, Egypt

[2]Professor Doctor in Mathematical statistics, faculty of graduate studies for statistical research, Cairo University, Egypt

[3]Associate Professor in Operations Research, faculty of graduate studies for statistical research, Cairo University, Egypt

Abstract: *The assembly line balancing problems have great importance in research and industry fields. They allow minimizing the learning aspects and guaranteeing a fixed number of products per day. This paper introduces a new problem that combines the multi-manned concept with the U-shaped lines with time and space constraints under uncertainty. The processing time of the tasks is considered as random variables with known means and variances. Therefore, chance-constraints appear in the cycle time constraints. In addition, each task has an associated area, where the assigned tasks per station are restricted by a total area. The proposed algorithm for solving the problem is a stochastic local search algorithm. The parameter levels of the proposed algorithm are optimized by the Taguchi method to cover the small, medium, and large-sized problems. Well-known benchmark problems have been adapted to cover the new model. The computational results showed the importance of the new problem and the efficiency of the proposed algorithm.*

Keywords: **U-shaped assembly line balancing problem; multi-manned assembly line balancing problem; The Taguchi method; chance-constrained programming; metaheuristics; stochastic local search.**

1. Introduction

The assembly lines are of great importance in the industry. They alternate the traditional way of assembling all tasks of a product by only one operator to assembling them in a set of stations, each of which has either one operator or more. The assembly line balancing problem is related to optimizing the assignment of the tasks with respect to some constraints. The simplified assumptions of the problem consider that the constraints are related to assigning each task in only one station, each station has a total processing time less than or equal to the cycle time, which is the total time spanned between two sequential products, and each task should be assembled after its predecessors. In most literature, the objective functions of this problem were to optimize either the number of stations, the cycle time, or the line efficiency. In addition, the problem in literature has been classified into simple assembly line balancing problems (SALBP), which is related to the simplified assumptions, and general assembly line balancing problems (GALBP). GALBP problems are related to adding some practical constraints to SALBP. This paper considers one of GALBP, which contains constraints related to stochastic processing times of the tasks, the form of the line, which is U-shaped, the area related to each task, and the multi-manned concept. There are two objective functions of the problem in this paper. The first one is to minimize the number of stations and the second one is to minimize the number of operators. The proposed algorithm for solving the problem is a stochastic local search algorithm, which is a

metaheuristic approach that uses random walk solutions to cover the most areas in the solution space and a local search around each random walk to guarantee convergence. The paper is organized as follows. The second section shows a literature review that covers some papers published in both U-shaped and multi-manned assembly line balancing problems. The third section shows the mathematical model of the proposed problem. The fourth section shows the proposed algorithm. The fifth section presents the computational results. Finally, the sixth section is the conclusion.

2. Literature Review

This paper discusses solving the multi-manned U-shaped assembly line balancing problem in case that the processing times of the tasks are random variables with known means and variances, and each task has an associated area. Therefore, it combines two types of assembly line balancing problems and deals with them under uncertainty. In this section, the literature review shows that this form of the problem, to the best of our knowledge, has not been investigated before and the stochastic local search seems that it was not developed to solve the assembly line balancing problems.

2.1. U-shaped assembly line balancing problem

The first work in the U-shaped assembly line balancing problems began by Miltenburg and Wijngaard [1]. They modeled the problem and developed a heuristic approach for solving it. Ajenblit and Wainwright [2] developed a genetic algorithm for solving the problem. They aimed to minimize the number of stations. Their research provided six different assignment algorithms. Nakade and Ohno [3] showed the first upgrade of the problem by including a worker assignment procedure. Their main objective was to minimize the cycle time by optimally allocating workers by using a heuristic algorithm. Gokcen and Agpak [4] developed a goal programming that minimizes the deviation of three goals, which are related to the number of stations, cycle time, and the number of tasks per station. They aimed to find a satisfactory solution rather than finding an optimal solution due to the conflicting goals. Through their paper, they developed the first multi-criteria decision-making approach for the U-shaped assembly lines. Baykasoglu [5] developed a simulated annealing approach to maximize the smoothness index and to minimize the number of stations. As it appears from his paper, there is a new objective function has been included in his formulation, which is to maximize the smoothness index besides minimizing the number of stations. Kara et al. [6] applied a binary fuzzy goal programming approach for both straight and U-shaped assembly line balancing problems. Simaria et al. [7] produced a new form of the problem that is related to meeting the demand variations. They aimed to minimize the number of workers according to many scenarios. Bagher et al. [8] discussed the stochastic nature of the processing time of the tasks. Therefore, they showed that the processing times of the tasks are random variables with means and variances. So, the cycle time constraints were formulated as chance-constraints, such as existed herein. They developed an algorithm that combines a computer method for sequencing operations for assembly lines, task assignment heuristic rules, and imperialist competitive algorithm. Kara et al. [9] developed an integer programming formulation the deals with a multi-objective model, where the model is to minimize the total cost associated with work station utilization, assistant, assignment, and equipment allocation.

Agpak and Yegul [10] have formulated the two-sided and U-shaped assembly line balancing problem in one mathematical model and solved the new problem by using GAMS/CPLEX-12 mathematical programming software package. They aimed to minimize the number of stations and positions. Hamzadayi and Yildiz [11] developed a simulated annealing approach for balancing mixed-model U-lines. Their objective

function was to minimize the number of stations. Hazır and Dolgui [12] formulated a robust optimization model for the U-shaped assembly line problem. They considered that the processing times of the tasks are represented as intervals, then they solved the problem by iterative approximation algorithm. Alavidoost et al. [13] developed an interactive fuzzy programming approach for minimizing the number of stations and the cycle time. They considered that the processing times of the tasks are triangular fuzzy numbers. Alavidoost et al. [14] formulated the U-shaped assembly line balancing problem with fuzzy processing time and solved it using genetic algorithm. They aimed to optimize the line efficiency and the percentage of idleness. Aydoğan et al. [15] developed particle swarm optimization for solving the U-shaped assembly line balancing problem in case that the processing times are stochastic. Zhang et al. [16] considered that the processing time of the tasks is related to the workers. So, they took into consideration the worker assignment. Their main objective is to minimize the cycle time. They developed migrating birds optimization algorithm for solving the problem. Zhang et al. [17] developed a multi-objective Jaya algorithm for solving UALBP considering maintenance scenarios. Chutima and Jirachai [18] addressed multi-model line balancing of parallel U-shaped assembly lines. They aimed to optimize multi-objectives, which are task un-relatedness, the variation of workload, and the number of stations. The developed algorithm is a hybridized multi-objective evolutionary optimization based on decomposition with biogeography-based optimization.

2.2. Multi-Manned assembly line balancing problem

The multi-manned assembly line balancing problem is concerned with assigning at least one operator to each station with taking into account sequencing constraints. This section shows some literature related to the multi-manned assembly line balancing problem.

Fattahi et al. [19] developed a mathematical model as well as an ant-colony optimization algorithm for solving the multi-manned assembly line balancing problem. Roshani et al.[20] developed a simulated annealing algorithm for solving the assembly line balancing problem with multi-manned stations. They aimed to maximize the line efficiency and smoothness index. Kellegöz and Toklu [21] proposed a priority rule-based constructive heuristic for solving the parallel multi-manned assembly line balancing problem. They used a genetic algorithm to improve their solutions found by their proposed heuristic. Hamid and Mustafa Yilmaz [22] considered load-balancing constraints in the assembly line balancing problem with multi-manned stations. They aimed to minimize the number of workers and stations. Kellegöz [23] proposed a Gannt-based heuristic method for solving the multi-manned assembly line balancing problem. The proposed heuristic in this paper has been improved by using simulated annealing. Roshani and Giglio [24] proposed a mathematical model for the multi-manned assembly line balancing problem. They aimed to minimize the cycle time and the number of workers. Their developed approach for solving the problem is simulated annealing. Chen [25] proposed a simulated annealing that aimed to minimize the number of stations and operators in the multi-manned assembly line balancing problem.

Chen et al. [26] proposed a mixed-integer programming model for the resource-constrained multi-manned assembly line balancing problem. They proposed a genetic algorithm to minimize the number of stations and operators. Michels et al. [27] developed a benders' decomposition algorithm for solving the multi-manned assembly line balancing problem. They aimed to minimize the number of workers as a primary objective and the number of stations as secondary. Şahin et al. [28] developed a mixed-integer linear programming model and particle swarm for solving resource investment

and balancing multi-manned assembly lines. They aimed to minimize the total cost. Lopes et al.[29] proposed station frontiers for multi-manned lines. They developed a mixed-integer programming model and model-based heuristic for solving the problem. Yilmaz et al. [30] proposed a tabu search algorithm for solving the multi-manned assembly line balancing problem. They aimed to minimize the number of workers and stations. Michels et al. [31] developed decomposition techniques and bender's cuts for minimizing the cycle time of multi-manned assembly line balancing problems.

3. The Mathematical Model

This section contains a mixed-integer programming formulation of the chance-constrained multi-manned U-shaped assembly line balancing problem.

3.1. Notations

$i = \{1, \dots, n\}$	The set of tasks
$j = \{1, \dots, m\}$	The set of stations
$k = \{1, \dots, l\}$	The set of operators
ct	Cycle time
A	The total area
k_{max}	The maximum number of operators in any station
t_i	The processing time of task i (random variable)
$E(t_i)$	The expected processing time of task i
$Var(t_i)$	The variance of task i
a_i	The area associated to task i
$IP(i)$	The immediate predecessors of task i
$IS(i)$	The immediate successors of tasks i

3.2. Decisions Variables

$$x_{ijk} = \begin{cases} 1, if\ task\ i\ assigned\ to\ operator\ k\ in\ station\ j \\ 0, otherwise \end{cases}$$

$$y_j = \begin{cases} 1, if \sum_{i=1}^{n} x_{ijk} > 0 \\ 0, otherwise \end{cases}$$ The opened station decision variable

$$R_k = \begin{cases} 1, if \sum_{i=1}^{n} x_{ijk} > 0 \\ 0, otherwise \end{cases}$$ The operator assignment decision variable

$$P_i = \begin{cases} 1, if \sum_{j=1}^{m} j\, x_{ijk} \geq \sum_{j=1}^{m} j\, x_{hjk}, \forall h \in IP(i) \\ 0, otherwise \end{cases}$$ The immediate predecessors assignment decision variable

$$S_i = \begin{cases} 1, if \sum_{j=1}^{m} j\, x_{ijk} \geq \sum_{j=1}^{m} j\, x_{hjk}, \forall h \in IS(i) \\ 0, otherwise \end{cases}$$ The immediate successors assignment decision variables

3.3. The Objective Function

$$Min. \sum_{j=1}^{m} y_j - \frac{1}{\sum_{k=1}^{l} R_k} \quad (1)$$

3.4. The constraints

$$\sum_{j=1}^{m} x_{ijk} = 1, \forall i = \{1, \dots, n\} \quad (2)$$

$$P\left(\sum_{i=1}^{n} t_i \, x_{ijk} \leq k_{max} \, ct\right) \geq \alpha, \forall j = \{1, \dots, m\} \quad (3)$$

$$\sum_{i=1}^{n} a_i \, x_{ijk} \leq A, \forall k = \{1, \dots, l\} \quad (4)$$

$$P_i + S_i \geq 1, \forall i = \{1, \dots, n\} \quad (5)$$

$$P\left(\sum_{i=1}^{n} t_i \, x_{ijk} \leq \, ct\right) \geq \alpha, \forall k = \{1, \dots, l\} \quad (6)$$

$$P\left(t_i x_{ijk} + \sum_{g \in iP(i)} t_g x_{gjk} + \sum_{h \in iS(i)} t_h x_{hjk} \leq ct\right) \geq \alpha, \forall j = \{1, \dots, m\} \quad (7)$$

The objective function (1) seeks to minimize two objectives, which are the number of stations as a primary objective and the number of operators as a secondary. The set of constraints (2), which are the assignment constraints, ensures that each task must be assigned in only one station. The set of chance-constraints (3), which are the stations' cycle time constraints, shows that each station contains tasks that their processing times are less than or equal to the cycle times of its operators under chance probability less than or equal α. The set of constraints (4), which are the space constraints, ensures that each operator should implement a set of tasks that their areas are less than or equal to a predetermined total area A. The set of constraints (5), which are the predecessors and successors' constraints, shows that each task in any station must be assigned after either its immediate predecessors or immediate successors. The set of constraints (6), which are the cycle time constraints of operators, ensures that each operator implements a set of tasks that their total processing time is less than or equal to the cycle time and they are also chance-constraints that restricted by probability less than or equal α. The set of constraints (7), which are the sequence constraints, ensures that if any immediate predecessors or immediate successors of a task are included in the same station of such task, then the sum of their processing times plus the task processing time must not exceed the cycle time. This set of constraints are also chance-constraints that are restricted by probability less than or equal α.

As aforementioned, the mathematical model contains some set of chance-constraints that need to be converted into a deterministic form. The assumption herein is to consider the processing times of the tasks as normally distributed random numbers that have known means and variances. Taha [32] mentioned in his book how to convert the chance-constraints to non-linear deterministic constraints. The procedures mentioned in this book can be used with the set of constraints (3), (6), and (7), where the results replace these set of constraints, each of which respectively, as follows:

$$\sum_{i=1}^{n} E(t_i) \, x_{ijk} + K_\alpha \sqrt{\sum_{i=1}^{n} Var(t_i) x_{ijk}} \leq k_{max} \, ct, \forall j = \{1, \dots, m\}, where \; K_\alpha \; is \; the \; standard \; normal \; value \; of \; \alpha \quad (8)$$

$$\sum_{i=1}^{n} E(t_i) \, x_{ijk} + K_\alpha \sqrt{\sum_{i=1}^{n} Var(t_i) x_{ijk}} \leq \, ct, \forall k = \{1, \dots, l\} \quad (9)$$

$$E(t_i)x_{ijk} + \sum_{g \in iP(i)} E(t_g)x_{gjk} + \sum_{h \in iS(i)} E(t_h)x_{hjk} + K_\alpha \sqrt{Var(t_i)x_{ijk} + \sum_{g \in iP(i)} Var(t_g)x_{gjk} + \sum_{h \in iS(i)} Var(t_h)x_{hjk}} \leq ct, \forall j = \{1, \dots, m\} \quad \textbf{(10)}$$

4. The Proposed Algorithm

The proposed algorithm for solving the problem depends on the concept of stochastic local search algorithms (SLS). Those algorithms begin by generating a random walk, which represents a solution in the solution space. Then a local search is to be utilized to find better solutions around such random walk solution. The iterative procedure of the algorithm allows exploring more areas in the solution space using those random walks and the local search helps to find the local best solutions for each area. The global best solution is the best solution found per all iterations. The random walk solution in the proposed algorithm is to be created by generating a random sequence of the tasks, where such random sequence represents the priority structure that will be used as a base of a heuristic procedure. Thus, each task in the priority structure will be tested if it confirms the problem constraints, and if it does, then it will be assigned in the opened station. In the local search, the random walk priority structure will be mutated by randomly swapping a percentage of its first tasks with the remaining tasks of the structure. Then, the heuristic procedure will be applied again. The algorithm contains a fixed number of random walks, mutation rate, and local search solutions, where those parameters are optimized later on using The Taguchi method. Figure 1 shows the flowchart of the proposed algorithm.

For farther illustration of the flowchart, the pseudo code of the proposed algorithm is as follows:

```
Begin
    Initialize the algorithm's parameters (N, MR, LN)
    Set f(G) = ∞ and i = 1
    While i ≤ N
        Generate random priority structure (RS_i)
        Generate random walk solution using heuristic based on RS_i
        Set j = 1, f(L) = ∞
        While j ≤ LN
            Set mutated structure(MS_j) = RS_i after randomly swap its first MR tasks
            Generate L_j solution using heuristic based on MS_j structure
            if f(L_i) < f(L) then f(L) = f(L_i) and L = L_i
            j = j + 1
            End
        if f(L) < f(G) then f(G) = f(L) and G = L
        i = i + 1
    End
End
```

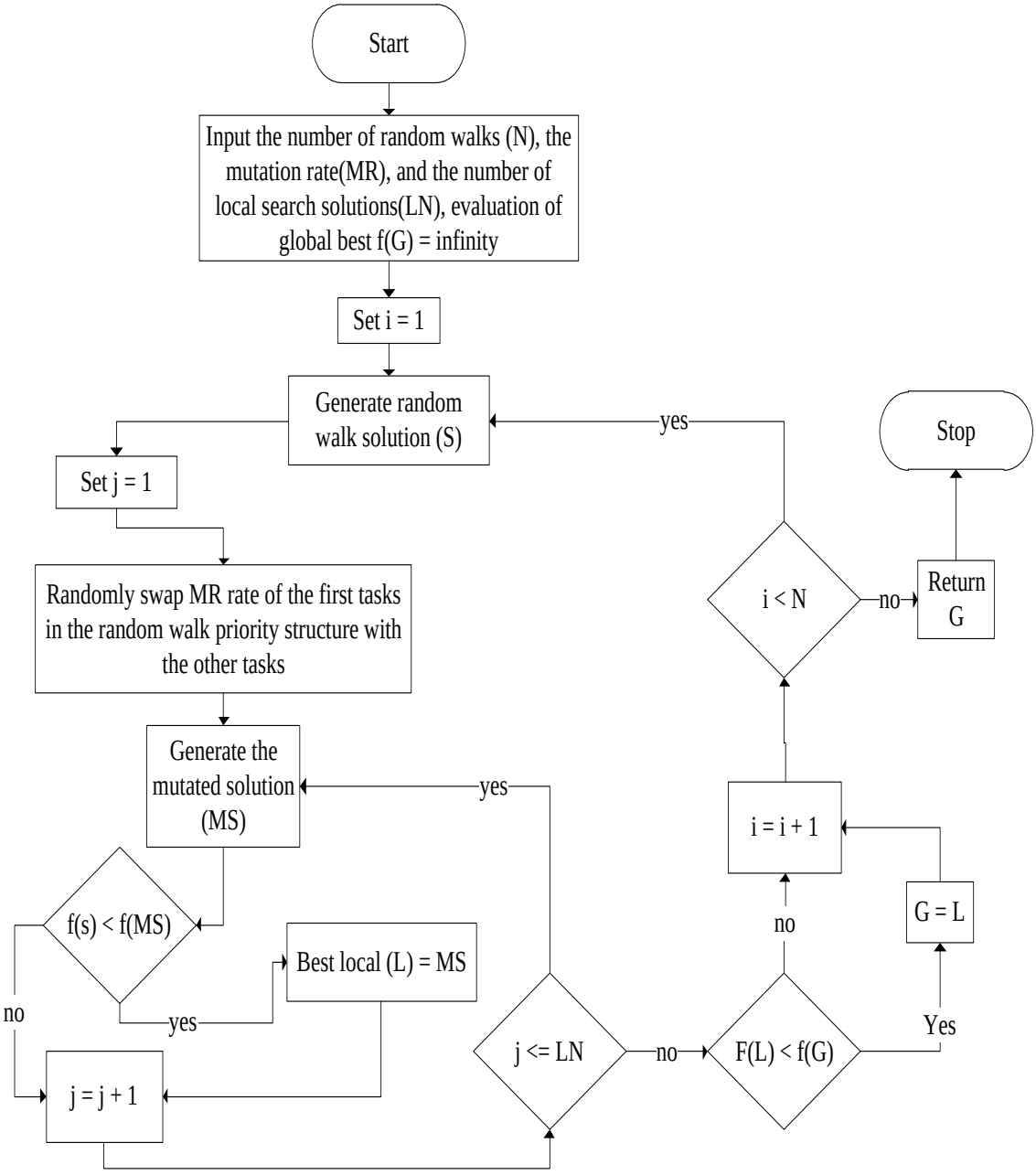

Figure 1: The flowchart of the proposed algorithm

4.1. Experimental design

The proposed SLS algorithm has three parameters, which are the number of random walks, the mutation rate, and the number of local search solutions. In order to optimize the parameters of the algorithm, The Taguchi method is used as follows. The proposed levels of the first parameter are 10, 15, and 20 random walks. The proposed levels of the second parameter are 0.05, 0.15, and 0.20. The proposed levels of the third parameter are 20, 50, and 80 local solutions. The problems found in https://assembly-line-balancing.de/ differ in their sizes from 7 tasks to 297. Those problems can be classified into three categories, small, medium, and large. Table 1 shows the selected problems for calibration, where they are two problems from each category. The proposed response to evaluate the output of the trials is the relative percentage deviation (RPD). The equation of RPD is as follows:

$$RPD = \frac{f(s) - f(s_{best})}{f(s_{best})} \times 100 \quad \textbf{(11)}$$

Both $f(s)$ and $f(s_{best})$ are calculated by subtracting the inverse of CPU times of the algorithm from the value of the objective function. $f(s_{best})$ represents the best solution of all trials.

In the Taguchi approach, an orthogonal array is used to construct the required trials that should be implemented to optimize the parameter levels. According to the selected levels, which are 3 levels for each parameter, the required experiments in the full factorial design are 27 experiments, which are as in Table 2. After implementing the trials with respect to the settings included in each row in Table 2, the results of each problem size have been compared by using the analysis of variance (ANOVA). Table 3 shows the analysis of variance, which indicates that the null hypothesis is rejected. Therefore, Tukey pairwise comparison has been applied to find which means differ. Table 4 shows the grouping information using the Tukey method and Figure 2 shows the interval plot.

Table 1: The selected problems for experimental design

Name	Problem size	Minimal task time	Maximal task time	Cycle time	Problem class
Buxey	29	1	25	27	Small
Gunther	35	1	40	44	Small
Arcus1	83	233	3691	3786	Medium
Mukherje	94	8	171	176	Medium
Arcus2	111	10	5689	5755	Large
Barthold	148	3	383	403	Large

Table 2: The required orthogonal array for experimental design

Number of Random walks	Mutation rate	Number of local solutions
5	0.05	20
5	0.05	20
5	0.05	20
5	0.1	50
5	0.1	50
5	0.1	50
5	0.15	80
5	0.15	80
5	0.15	80
10	0.05	50
10	0.05	50
10	0.05	50
10	0.1	80
10	0.1	80
10	0.1	80
10	0.15	20
10	0.15	20
10	0.15	20
15	0.05	80
15	0.05	80

Number of Random walks	Mutation rate	Number of local solutions
15	0.05	80
15	0.1	20
15	0.1	20
15	0.1	20
15	0.15	50
15	0.15	50
15	0.15	50

Table 3 Analysis of variance of the RPDs experiments

Source	DF	Sum of squres	Mean squares	F-Value	P-Value
RPDs	5	4.255	0.85103	25.04	0
Error	156	5.301	0.03398		
Total	161	9.556			

Table 4 The grouping information of Tukey method

Problem	Mean	Grouping	Problem size
Barthold	0.4515	A	Large
Arcus2	0.3308	A	Large
Mukherje	0.1661	B	Medium
Arcus1	0.1009	B C	Medium
Gunther	0.016	C	Small
Buxey	0.01335	C	Small

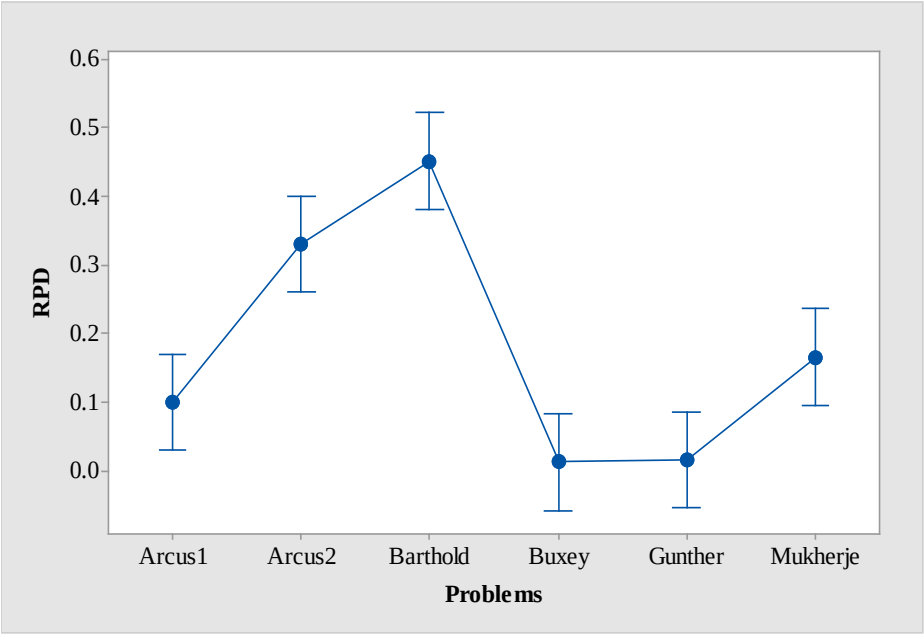

Figure 2 Interval plot of Tukey comparison method for RPD experiments

According to the grouping information, each problem size may have different parameter levels of the proposed algorithm. Therefore, for each problem size, The Taguchi method has been applied to optimize the parameter levels. Figure 3, Figure 4, and Figure 5 show the main effects plot for each problem size.

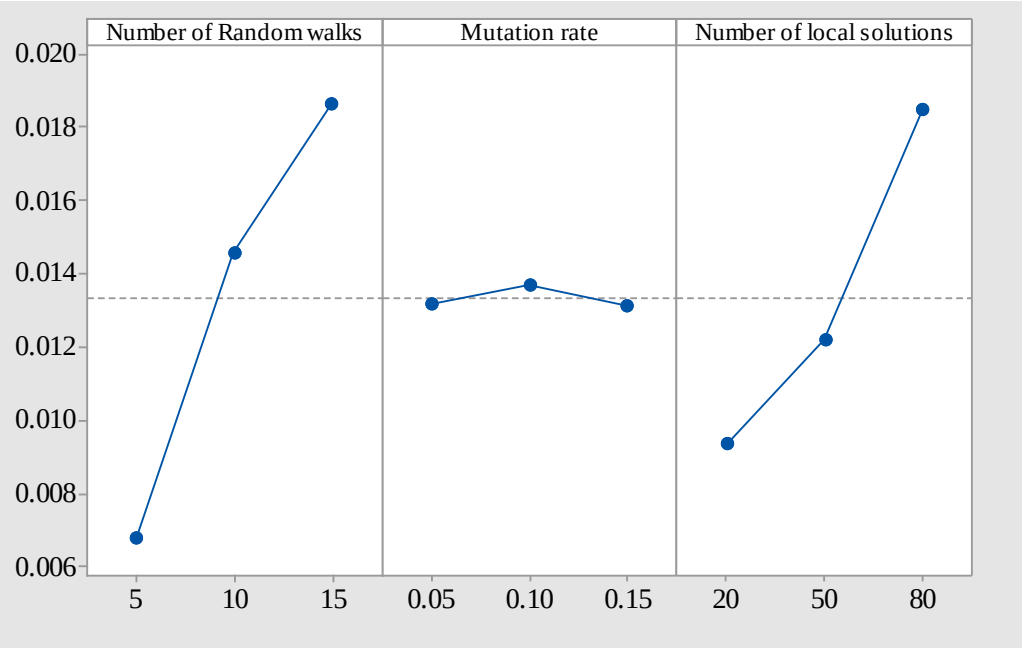

Figure 3 Main effects plot for small-sized problems

Figure 4 Main effects plot for medium-sized problems

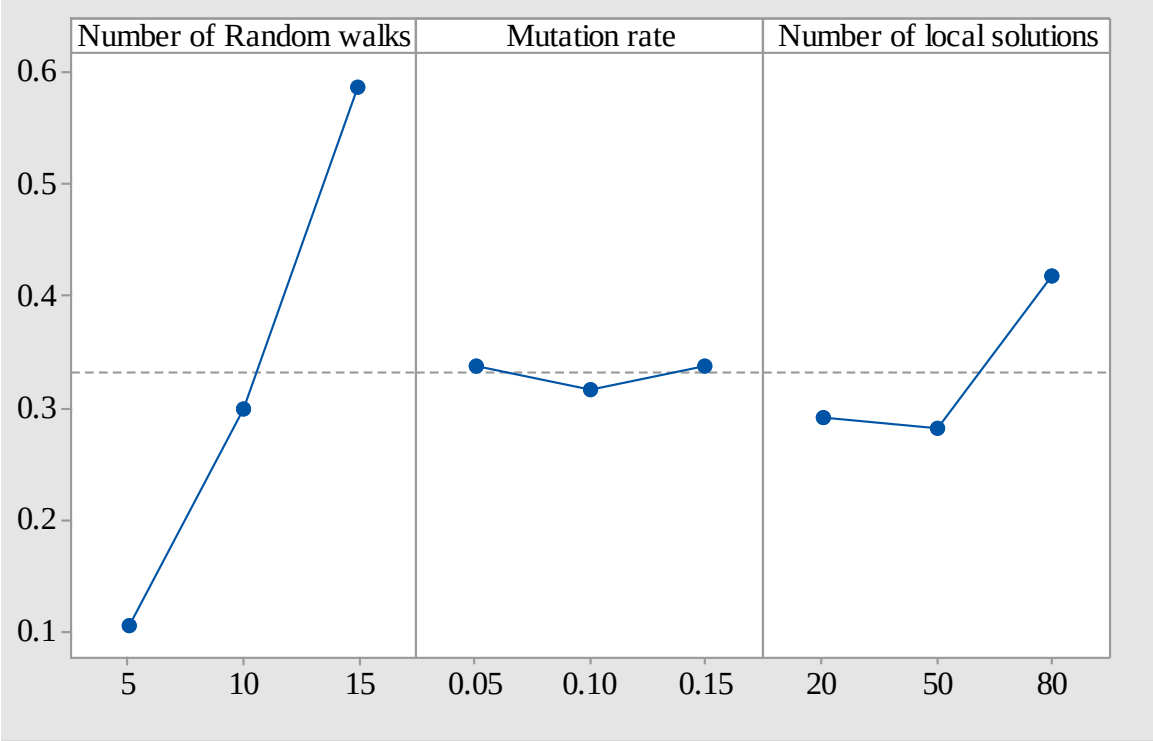

Figure 5 Main effects plot for large-sized problems

The main effects plots show that for both the small and medium-sized problems, the optimized parameter levels are 5 random walks, either using 0.05 or 0.15 as mutation rate and 20 local solutions. In the case of the large-sized problems, the optimized parameter levels are 5 random walks, 0.10 as mutation rate, and 50 local solutions.

4.2. Illustrative Example

The illustrative example shows the steps of the proposed algorithm using the optimized parameter levels. It contains the Jackson problem, which is one of the benchmark problems in https://assembly-line-balancing.de. The problem size is 11 tasks and it can be represented as a precedence graph network as shown in Figure 6. The cycle time is 7, the total area is 14, the maximum number of operators per station is 2, and the chance probability is 0.95. The first step of the algorithm is to generate a random priority structure (RS) of the problem tasks, which is used to create a random walk solution. The first priority structure (RS_1) can be shown as follows:

$$RS_1 = \{1,5,11,10,2,6,9,4,8,7,3\}$$

The heuristic solution obtained from RS_1 is in Table 5. It shows that the number of stations is 5 and the number of operators is 9. Now, RS_1 will be mutated by swapping a 0.05 percentage of the left first tasks in RS_1, where only task 10 will be swapped randomly with one of the rest of the tasks. After mutatingRS_1, a new solution will be obtained by repeating the same heuristic procedure. The mutation process will be done to produce 20 local solutions, and then a new iteration will begin to produce a new

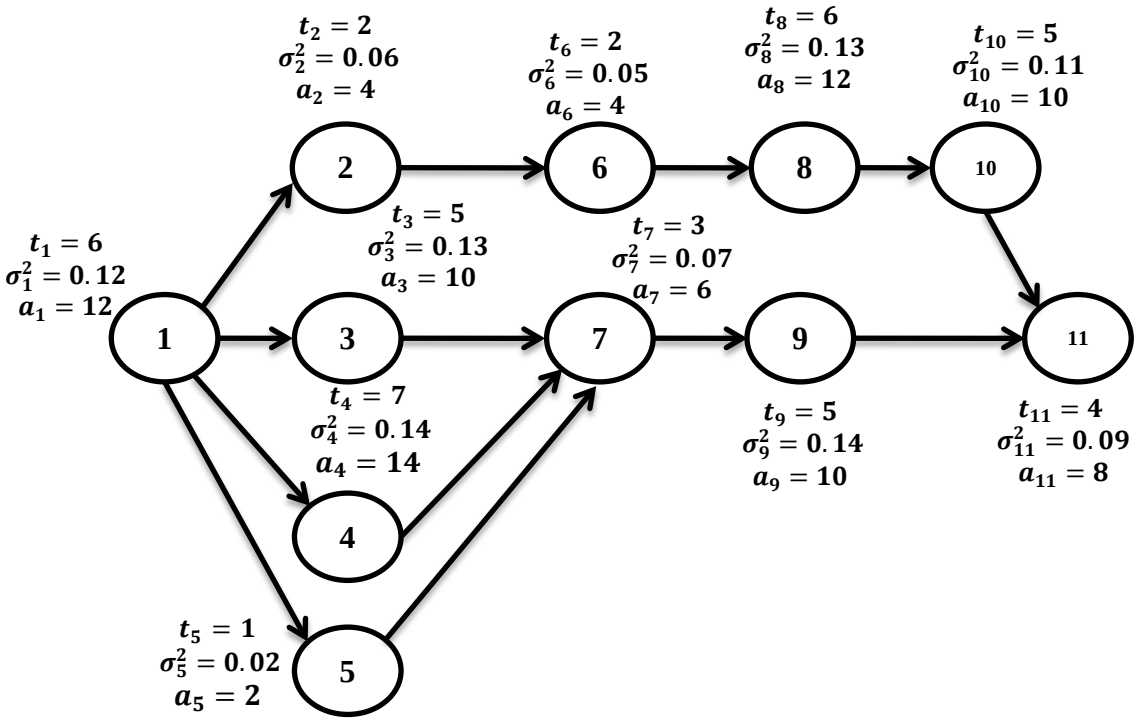

Figure 6 Precedence graph of Jackson problem

random walk. For each iteration, the best local solution will be compared with the best global solution and it will replace it if it is better. As aforementioned, the number of random walks is 5. Therefore, Table 6 shows the final solution after implementing 5 iterations. The solution shows that the number of stations is 4 and the number of operators is 8.

Table 5: The first random walk solution

Task	Processing time	Area	Station	Operator	Completion time
1	6	12	1	1	6
5	1	2	1	1	7
11	4	8	1	2	4
10	5	10	2	3	5
2	2	4	2	3	7
9	5	10	2	4	5
6	2	4	3	5	2
4	7	14	3	6	7
8	6	12	4	7	6
7	3	6	4	8	3
3	5	10	5	9	5

Table 6: The final solution (The global best)

Task	Processing time	Area	Station	Operator	Completion time
11	4	8	1	1	4
1	6	12	1	2	6
5	1	2	1	2	7
4	7	14	2	3	7
3	5	10	2	4	5
2	2	4	2	4	7
10	5	10	3	5	5
6	2	4	3	5	7
9	5	10	3	6	5
8	6	12	4	7	6
7	3	6	4	8	3

5. Computational results

The computational results section shows the results of implementing the proposed algorithm on 71 problems from the benchmark found in https://assembly-line-balancing.de after adapting them to fit the requirements of the new model. The benchmark problems do not have areas, expected values of the processing times, and variances. Therefore, the area associated with each task is assumed to be 2 times the processing time and the total area is assumed to be 2 times the cycle time. The expected processing times are assumed to be the same as the deterministic processing times of the benchmarks and the variances are found by subtracting each processing time from the cycle time and divide the result by 1000. The maximum number of operators is 2. As far as known, the proposed model in this paper has not been discussed before in research. Therefore, to show the importance of the new model, the comparison is done with the deterministic multi-manned assembly line balancing problem. Table 7 shows a comparison between the values found by the proposed algorithm for solving the chance-constrained multi-manned U-shaped assembly line balancing problem with time and space and some other algorithms that were used to solve the deterministic multi-manned assembly line balancing problem. Such algorithms are the ant-colony optimization (ACO) algorithm by Fattahi [19] and the simulated annealing (SA) by Roshani and Giglio [24]. The proposed algorithm is coded using python programming in a PC with 2.93 GHz Core 2 Duo CPU and 2 GB rams. The CPU times are compared in seconds between the proposed SLS and SA.

Table 7 Computational results

Problem	Size	Cycle time	ACO		SA			SLS		
								$\alpha = 0.95$		
			No. of operators	No. of stations	No. of operators	No. of stations	CPU (S)	No. of operators	No. of stations	CPU (S)
Merten	7	6	6	3	6	3	0.15	6	3	0.20
		7	5	3	5	3	0.14	5	3	0.32
		8	5	3	5	3	0.15	5	3	0.16
		10	3	3	3	3	0.14	3	***2***	0.29

Problem	Size	Cycle time	ACO		SA			SLS		
								$\alpha = 0.95$		
			No. of operators	No. of stations	No. of operators	No. of stations	CPU (S)	No. of operators	No. of stations	CPU (S)
Bowman	8	15	2	2	2	2	0.12	2	***1***	0.10
		18	2	1	2	1	0.09	2	1	0.10
		17	5	5	5	5	0.18	5	***3***	0.18
		20	–	–	5	4	0.17	***4***	***2***	0.14
		21	5	4	5	4	0.19	***4***	***2***	1.37
		24	4	4	4	4	0.18	4	***2***	0.59
		28	3	2	3	2	0.17	3	2	0.14
		31	3	2	3	2	0.2	3	2	0.13
Jaeschke	9	6	8	6	8	6	0.26	8	***4***	0.21
		7	7	6	7	6	0.18	7	***4***	0.20
		8	6	5	6	5	0.19	6	***3***	0.19
		10	4	4	4	4	0.21	4	***2***	0.17
		18	3	2	3	2	0.2	3	2	0.15
Jackson	11	7	8	6	8	6	0.1	8	***4***	0.34
		9	6	4	6	4	0.39	6	***3***	0.27
		10	5	4	5	4	0.4	5	***3***	0.50
		13	4	3	4	3	0.25	4	***2***	0.43
		14	4	3	4	3	0.25	4	***2***	0.24
		21	3	2	3	2	0.28	3	2	0.19
Mansoor	11	45	5	3	5	3	0.23	5	3	0.22
		54	4	3	4	3	0.21	4	***2***	0.44
		63	3	2	3	2	0.26	3	2	1.05
		72	3	2	3	2	0.18	3	2	0.22
		81	3	2	3	2	0.25	3	2	0.21
Mitchell	21	14	8	7	8	7	0.92	8	***4***	4.85
		15	8	7	8	7	0.96	8	***4***	1.15
		21	***5***	5	***5***	5	0.9	6	***3***	0.55
		26	5	4	5	4	0.79	5	***3***	0.41
		35	3	3	3	3	0.85	3	***2***	2.13
		39	3	2	3	2	0.81	3	2	0.43
Heskia	28	138	***8***	5	***8***	5	32.3	9	5	1.38
		205	***5***	3	***5***	3	34.8	6	3	1.30
		216	5	3	5	3	29.6	5	3	1.48
		256	***4***	3	***4***	3	32.4	5	3	1.38
		324	4	2	4	2	23.9	4	2	1.29
		342	***3***	2	***3***	2	15	4	2	1.15
Sawyer	30	25	***14***	8	***14***	8	41.9	15	8	2.63
		27	***13***	8	***13***	8	40.4	14	***7***	1.20
		30	12	7	12	7	38.6	12	***6***	1.12
		36	10	6	10	6	37.1	10	***5***	2.22
		41	***8***	6	***8***	5	34.3	9	5	0.90
		54	7	4	7	4	33.7	7	4	1.04

Problem	Size	Cycle time	ACO		SA			SLS		
								$\alpha = 0.95$		
			No. of operators	No. of stations	No. of operators	No. of stations	CPU (S)	No. of operators	No. of stations	CPU (S)
		75	5	3	5	3	25.8	5	3	0.79
Kilbridge	45	57	***10***	6	***10***	6	81.5	11	6	1.57
		79	***7***	5	***7***	5	105.8	8	***4***	1.53
		92	***6***	4	***6***	4	135.2	7	4	1.45
		110	6	3	6	3	68.5	6	3	1.39
		138	***4***	3	***4***	3	80.5	5	3	1.84
		184	3	2	3	2	51.2	3	2	1.59
Tonge	70	176	***21***	20	***21***	19	193.5	24	***12***	3.54
		364	***10***	7	***10***	7	309.8	11	***6***	3.10
		410	9	6	9	5	291.2	9	5	20.60
		468	8	4	8	4	106.3	8	4	6.28
		527	7	4	7	4	277.7	7	4	2.67
Arcus	83	5048	***16***	11	***16***	11	369.8	17	***9***	2.90
		5853	14	10	14	9	405.5	14	***7***	2.64
		6842	12	8	12	8	352.2	12	***6***	2.32
		7571	11	7	11	7	278.5	11	***6***	2.48
		8412	10	6	10	6	354.6	10	***5***	2.85
		8998	9	6	9	6	336.3	9	***5***	2.61
		10,816	8	5	8	5	355.7	8	***4***	3.02
Arcus	111	5755	***27***	20	***27***	21	578.9	31	***16***	8.03
		8847	***18***	12	***18***	12	603.2	19	***10***	67.60
		10027	16	10	16	11	765.4	16	***9***	29.72
		10743	15	10	15	10	1235.6	15	***8***	39.74
		11378	***14***	9	***14***	9	738.9	15	***8***	7.26
		17067	***9***	6	***9***	6	1335.9	10	***5***	8.55

Despite having space constraints and uncertainty in the new problem, it can be shown from Table 7 that the new problem and the proposed SLS algorithm both prove efficiency. 40 problems have a reduction in the number of stations and 2 problems have a reduction in the number of operators where that can be shown as bold italic numbers under SLS columns. The main objective in the proposed model is to minimize the number of stations as a primary objective and to minimize the number of operators as a secondary objective. Therefore, 18 problems are better than the proposed SLS concerning the number of operators. Figure 7 shows the box plots of the CPU times for both SLS and SA algorithms. The figure shows a very low variability in the case of SLS when compared with SA. The median of SLS CPU times is 1.15 seconds, while the median of SA CPU times is 23.9 seconds. Therefore, although the low performance of the used PC, the proposed algorithm proves efficiency in terms of CPU times.

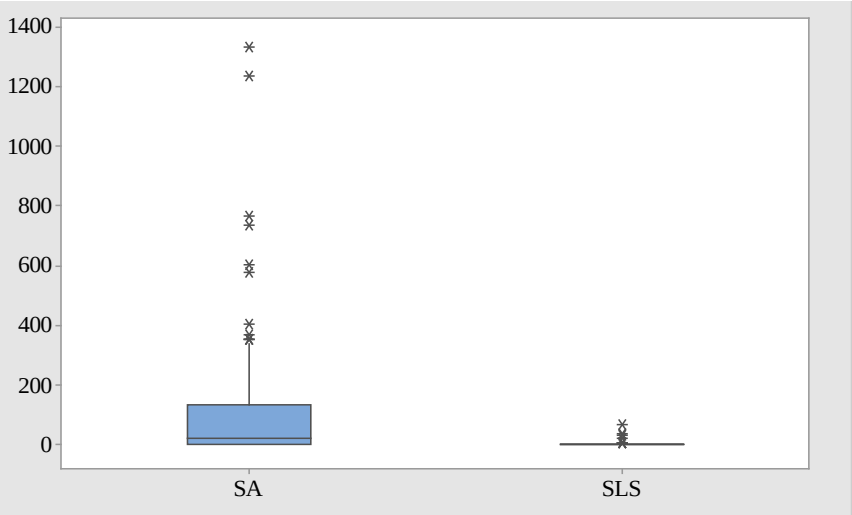

Figure 7 Box plots of CPU times

6. Conclusion

The use of the multi-manned concept helped to radically minimize the number of stations. The U-shaped lines helped for more flexibility in the assignment procedures, which may also minimize the number of stations. Therefore, after combining the multi-manned concept with the U-shaped assembly line balancing problem, a new mathematical model has been developed, which considers that the processing times are random variables with known means and variances, where that lead to a chance-constrains, and considers that each task has an area. A stochastic local search algorithm has been developed to solve the new problem and its parameters have been optimized by using the Taguchi method. The computational results show the efficiency of combining the multi-manned concept with the U-shaped lines and they show also the efficiency of the proposed stochastic local search algorithm.

References

[1] G. J. Miltenburg and J. Wijngaard, "U-line line balancing problem," *Manage. Sci.*, vol. 40, no. 10, pp. 1378–1388, 1994, doi: 10.1287/mnsc.40.10.1378.

[2] D. A. Ajenblit and R. L. Wainwright, "Applying genetic algorithms to the U-shaped assembly line balancing problem," *Proc. IEEE Conf. Evol. Comput. ICEC*, pp. 96–101, 1998, doi: 10.1109/icec.1998.699329.

[3] K. Nakade and K. Ohno, "Optimal worker allocation problem for a U-shaped production line," *Int. J. Prod. Econ.*, vol. 60, pp. 353–358, 1999, doi: 10.1016/S0925-5273(98)00145-5.

[4] H. Gökçen and K. Ağpak, "A goal programming approach to simple U-line balancing problem," *Eur. J. Oper. Res.*, vol. 171, no. 2, pp. 577–585, 2006, doi: 10.1016/j.ejor.2004.09.021.

[5] A. Baykasoğlu, "Multi-rule multi-objective simulated annealing algorithm for straight and U type assembly line balancing problems," *J. Intell. Manuf.*, vol. 17, no. 2, pp. 217–232, 2006, doi: 10.1007/s10845-005-6638-y.

[6] Y. Kara, T. Paksoy, and C. Ter Chang, "Binary fuzzy goal programming approach to single model straight and U-shaped assembly line balancing," *Eur. J. Oper. Res.*, vol. 195, no. 2, pp. 335–347, 2009, doi: 10.1016/j.ejor.2008.01.003.

[7] A. S. Simaria, M. Zanella De Sá, and P. M. Vilarinho, "Meeting demand variation using flexible U-shaped assembly lines," *Int. J. Prod. Res.*, vol. 47, no. 14, pp. 3937–3955, 2009, doi: 10.1080/00207540701871044.

[8] M. Bagher, M. Zandieh, and H. Farsijani, “Balancing of stochastic U-type assembly lines: An imperialist competitive algorithm,” *Int. J. Adv. Manuf. Technol.*, vol. 54, no. 1–4, pp. 271–285, 2011, doi: 10.1007/s00170-010-2937-3.

[9] Y. Kara, C. Özgüven, N. Yalçin, and Y. Atasagun, “Balancing straight and U-shaped assembly lines with resource dependent task times,” *Int. J. Prod. Res.*, vol. 49, no. 21, pp. 6387–6405, 2011, doi: 10.1080/00207543.2010.535039.

[10] K. Ağpak, M. F. Yegül, and H. Gökçen, “Two-sided U-type assembly line balancing problem,” *Int. J. Prod. Res.*, vol. 50, no. 18, pp. 5035–5047, 2012, doi: 10.1080/00207543.2011.631599.

[11] A. Hamzadayi and G. Yildiz, “A simulated annealing algorithm based approach for balancing and sequencing of mixed-model U-lines,” *Comput. Ind. Eng.*, vol. 66, no. 4, pp. 1070–1084, 2013, doi: 10.1016/j.cie.2013.08.008.

[12] Ö. Hazir and A. Dolgui, “A decomposition based solution algorithm for U-type assembly line balancing with interval data,” *Comput. Oper. Res.*, vol. 59, pp. 126–131, 2015, doi: 10.1016/j.cor.2015.01.010.

[13] M. H. Alavidoost, H. Babazadeh, and S. T. Sayyari, “An interactive fuzzy programming approach for bi-objective straight and U-shaped assembly line balancing problem,” *Appl. Soft Comput. J.*, vol. 40, pp. 221–235, 2016, doi: 10.1016/j.asoc.2015.11.025.

[14] M. H. Alavidoost, M. H. F. Zarandi, M. Tarimoradi, and Y. Nemati, “Modified genetic algorithm for simple straight and U-shaped assembly line balancing with fuzzy processing times,” *J. Intell. Manuf.*, vol. 28, no. 2, pp. 313–336, 2017, doi: 10.1007/s10845-014-0978-4.

[15] E. K. Aydoğan, Y. Delice, U. Özcan, C. Gencer, and Ö. Bali, “Balancing stochastic U-lines using particle swarm optimization,” *J. Intell. Manuf.*, vol. 30, no. 1, pp. 97–111, 2019, doi: 10.1007/s10845-016-1234-x.

[16] Z. Zhang, Q. Tang, D. Han, and Z. Li, “Enhanced migrating birds optimization algorithm for U-shaped assembly line balancing problems with workers assignment,” *Neural Comput. Appl.*, vol. 31, no. 11, pp. 7501–7515, 2019, doi: 10.1007/s00521-018-3596-9.

[17] Z. Zhang, Q. Tang, D. Han, and X. Qian, “An enhanced multi-objective JAYA algorithm for U-shaped assembly line balancing considering preventive maintenance scenarios,” *Int. J. Prod. Res.*, vol. 0, no. 0, pp. 1–20, 2020, doi: 10.1080/00207543.2020.1804639.

[18] P. Chutima and P. Jirachai, “Parallel U-shaped assembly line balancing with adaptive MOEA/D hybridized with BBO,” *J. Ind. Prod. Eng.*, vol. 37, no. 2–3, pp. 97–119, 2020, doi: 10.1080/21681015.2020.1735544.

[19] P. Fattahi, A. Roshani, and A. Roshani, “A mathematical model and ant colony algorithm for multi-manned assembly line balancing problem,” *Int. J. Adv. Manuf. Technol.*, vol. 53, no. 1–4, pp. 363–378, 2011, doi: 10.1007/s00170-010-2832-y.

[20] A. Roshani, A. Roshani, A. Roshani, M. Salehi, and A. Esfandyari, “A simulated annealing algorithm for multi-manned assembly line balancing problem,” *J. Manuf. Syst.*, vol. 32, no. 1, p. 238, 2013, doi: 10.1016/j.jmsy.2012.11.003.

[21] T. Kellegöz and B. Toklu, “A priority rule-based constructive heuristic and an improvement method for balancing assembly lines with parallel multi-manned workstations,” *Int. J. Prod. Res.*, vol. 53, no. 3, pp. 736–756, 2015, doi: 10.1080/00207543.2014.920548.

[22] H. Yilmaz and M. Yilmaz, “Multi-manned assembly line balancing problem with balanced load density,” *Assem. Autom.*, vol. 35, no. 1, pp. 137–142, 2015, doi: 10.1108/AA-05-2014-041.

[23] T. Kellegöz, “Assembly line balancing problems with multi-manned stations: a new mathematical formulation and Gantt based heuristic method,” *Ann. Oper. Res.*, vol. 253, no. 1, pp. 377–404, 2017, doi: 10.1007/s10479-016-2156-x.

[24] A. Roshani and D. Giglio, “Simulated annealing algorithms for the multi-manned assembly line balancing problem: minimising cycle time,” *Int. J. Prod. Res.*, vol. 55, no. 10, pp. 2731–2751, 2017, doi: 10.1080/00207543.2016.1181286.

[25] Y. Y. Chen, “A hybrid algorithm for allocating tasks, operators, and workstations in multi-manned assembly lines,” *J. Manuf. Syst.*, vol. 42, pp. 196–209, 2017, doi: 10.1016/j.jmsy.2016.12.011.

[26] Y. Y. Chen, C. Y. Cheng, and J. Y. Li, “Resource-constrained assembly line balancing problems with multi-manned workstations,” *J. Manuf. Syst.*, vol. 48, no. July, pp. 107–119, 2018, doi: 10.1016/j.jmsy.2018.07.001.

[27] A. S. Michels, T. C. Lopes, C. G. S. Sikora, and L. Magatão, “A Benders’ decomposition algorithm with combinatorial cuts for the multi-manned assembly line balancing problem,” *Eur. J. Oper. Res.*, vol. 278, no. 3, pp. 796–808, 2019, doi: 10.1016/j.ejor.2019.05.001.

[28] M. Şahin and T. Kellegöz, “A new mixed-integer linear programming formulation and particle swarm optimization based hybrid heuristic for the problem of resource investment and balancing of the assembly line with multi-manned workstations,” *Comput. Ind. Eng.*, vol. 133, no. May 2018, pp. 107–120, 2019, doi: 10.1016/j.cie.2019.04.056.

[29] T. C. Lopes, G. V. Pastre, A. S. Michels, and L. Magatão, “Flexible multi-manned assembly line balancing problem: Model, heuristic procedure, and lower bounds for line length minimization,” *Omega (United Kingdom)*, vol. 95, no. xxxx, 2020, doi: 10.1016/j.omega.2019.04.006.

[30] H. Yilmaz and M. Yilmaz, “A mathematical model and tabu search algorithm for multi-manned assembly line balancing problems with assignment restrictions,” *Eng. Optim.*, vol. 52, no. 5, pp. 856–874, 2020, doi: 10.1080/0305215X.2019.1618288.

[31] A. S. Michels, T. C. Lopes, and L. Magatão, “An exact method with decomposition techniques and combinatorial Benders’ cuts for the type-2 multi-manned assembly line balancing problem,” *Oper. Res. Perspect.*, vol. 7, no. April, p. 100163, 2020, doi: 10.1016/j.orp.2020.100163.

[32] H. Taha, *Operations Research an Introduction*, 10th ed. Harlow: Pearson Education Limited, 2017.