International Journal of Production Management and Engineering

https://doi.org/10.4995/ijpme.2021.16084
Received: 2021-08-10 Accepted: 2021-09-30

Solving Stochastic Multi-Manned U-shaped Assembly Line Balancing Problem Using Differential Evolution Algorithm

Mohammad Zakaraia [a1], Hegazy Zaher [a2], Naglaa Ragaa [a3]

[a] Faculty of graduate studies for statistical research, Cairo University,
5 Ahmed Zewail, Ad Doqi, Dokki, Giza Governorate, Egypt.
[a1] *zicooo82@gmail.com,* [a2] *hgsabry@gmail.com*, [a3] *naglaa777subkiii@yahoo.com*

Abstract:

The U-shaped assembly lines help to have more flexibility than the straight assembly lines, where the operators can perform tasks in both sides of the line, the entrance and the exit sides. Having more than one operator in any station of the line can reduce the line length and thereby affects the number of produced products. This paper combines the U-shaped assembly line balancing problem with the multi-manned assembly line balancing problem in one problem. In addition, the processing times of the tasks are considered as stochastic, where they are represented as random variables with known means and variances. The problem is formulated as a mixed-integer linear programming and the cycle time constraints are formulated as chance-constraints. The proposed algorithm for solving the problem is a differential evolution algorithm. The parameter of the algorithm is optimized using experimental design and the computational results are done on 71 adapted problems selected from well-known benchmarks.

Key words:
Metaheuristics, Differential evolution algorithm, U-shaped assembly line balancing problem, multi-manned assembly line balancing problem, Chance-constrained programming.

1. Introduction

The assembly lines play an important role in industry. They reduce the learning aspects by dividing the assembly work into a set of stations that move in some kind of transportation system, such as conveyer belt, and they help to produce products in a fixed time called the cycle time. The assembly line balancing problem is the problem that is related to optimizing the assignment of the tasks to the stations in order to achieve some specific objectives, such as minimizing the number of stations and minimizing the cycle time. The simplified assumptions of the problem consist of three set of constraints. The first set of constraints is the set of assignment constraints, which ensures that each task is assigned in only one station. The second set of constraints is the cycle time constraints, which ensures that the total processing time of any station doesn't exceed the cycle time. The third set of constraints ensures that each task has to be assigned after its predecessors. The problem in research is classified into two categories, which are the simple assembly line balancing problems (SALBP) that only cover the simplified assumptions, and the general assembly line balancing problems (GALBP) which contain some other constraints related to the practical relevancies.

The U-shaped assembly line balancing problem (UALBP) represents one of GALBP. Its additional practical constraints are related the shape of the line, where the modification here is done on the precedence constraints by considering the successors beside the predecessors in the assignment procedure. The reason of having the successors in the assignment procedure is the shape of the line

To cite this article: Zakaraia, M., Zaher, H., Ragaa, N. (2022). Solving Stochastic Multi-Manned U-shaped Assembly Line Balancing Problem Using Differential Evolution Algorithm. *International Journal of Production Management and Engineering, 10*(1), 13-22. https://doi.org/10.4995/ijpme.2022.16084

allows to have the operators inside the line, where they can perform tasks either in the entrance or the exist sides of the line. The multi-manned assembly line balancing problem (MALBP) is another form of GALBP. It adds another set of constraints to the problem by considering the sequencing of the tasks that may restrict assign tasks to the additional operators in the same station. The contribution of this paper is that it combines both of UALBP and MALBP in one problem and presents a new mixed-integer programming model for such new problem. In addition, the problem is solved under uncertainty by having the stochastic processing times of the tasks. Therefore, the mathematical model shows the cycle time constraints as chance-constraints. Due to the combinatorial nature of the problem that makes it one of the NP-hard problems, the selected approach for solving it is an efficient metaheuristic called differential evolution algorithm (DE).

The paper is organized as follows. The second section shows a literature review that covers some of word presented in both of UALBP and MALBP. The third section presents the mathematical model of the new problem. The fourth section illustrates the developed DE for solving the problem. The fifth section shows a numerical example. The sixth section discusses the parameters of the proposed DE and optimizing its parameters using design of experiments (DOE). Eventually, the seventh section is the conclusion.

2. Literature Review

This paper discusses a combination between UALBP and MALBP. Therefore, this section covers some the previous works which are presented in both problems.

The first work in UALBP is presented by Miltenburg and Wijngaard (1994). They showed the advantages of using UALBP instead of using SALBP. UALBP in research can be classified into three categories according to objective functions: type-1, type-2, and type-E (Rabbani, Kazemi, and Manavizadeh, 2012). Type-1 is concerned with minimizing the number of stations for a given cycle time (Yilmaz et al., 2020). Type-2 objective is to minimize the cycle time with a given number of stations. Type-E is to maximize the line efficiency when the number of stations and the cycle time are unknown (Oksuz, Buyukozkan, and Satoglu, 2017). This paper considers multi-objectives for the new problem. The first objective is to minimize the number of stations and the second is to minimize the number of operators. The reason of having the number of operators as an objective is due to having the multi-manned concept of MALBP. In terms of minimizing the number of stations, there are a lot of papers that were presented in UALBP. For example, Ajenblit and Wainwright (1998) developed an ordered-based genetic algorithm for solving UALBP type-1 (UALBP-1). Scholl and Klein (Scholl and Klein, 1999) proposed a branch and bound procedure for solving many types of UALBP. Gökçen et al. (2006) presented a shortest route formulation for UALBP. Sabuncuoglu et al. (2009) developed ant colony optimization for solving UALBP. Kara et al. (2011) presented a resource dependent mathematical model for UALBP. Hamzadayi and Yildiz (2012) presented a genetic algorithm for solving UALBP in case of having parallel stations and mixed models. Hamzadayi and Yildiz (2013) developed a simulated annealing for solving UALBP in case of assembling mixed models. Jayaswal and Agarwal (2014) proposed a simulated annealing approach the resource dependent UALBP. Kucukkoc and Zhang (2015) proposed a hybrid design for the assembly line balancing problem that combines UALBP with the parallel assembly line balancing problem. They developed a heuristic procedure for solving such hybrid design. Fathi et al. (2016) developed a simulated annealing based proposed heuristic for solving UALBP. Li et al. (2017) presented a rules-based heuristic for solving UALBP. Sresracoo et al. (2018) developed DE algorithm for solving UALBP-1. Nourmohammadi et al. (2019) proposed a water flow inspired algorithm for solving UALBP. Zhang and Xu (2020) considered the energy cost in objective functions and solved UALBP using an improved flower pollination algorithm. Yılmaz (2020a) produced a robust optimizatoin for the U-shaped assembly line balancing problem with worker assignment in case that the processing times of the tasks are uncertain. Ö. F. Yılmaz (2020b) developed a mathematical model for an integrated bi-objective U-shaped assembly line balancing problem that considers heterogenity inhert of workers. He aimed to minimize the operational cost and workload balance. Li et al. (2021) developed an enhanced beam search heuristic for solving both of type-1 and type-2 of UALBP.

The first mathematical model of MALBP showed up in (Fattahi, Roshani, and Roshani, 2011a). They solved the problem using ant colony optimization algorithm. Fattahi et al. (2011b) developed an improved simulated annealing for solving MALBP. Their objectives are to minimize the number of

stations and to maximize the line efficiency. Kellegöz and Toklu (2015) proposed a genetic algorithm for solving MALBP. Their objective is to minimize the number of stations. Kellegöz (2017) another mathematical model for MALBP. He proposed a simulated annealing that uses a Gantt-based heuristic for solving the problem. Michels (2018) proposed genetic algorithm for solving MALBP. His objective is to minimize the costs per production unit. Michels et al. (2019) presented a mixed integer programming model for MALBP that can be solved using Benders' Decomposition Algorithm. Abidin Çil and Kizilay (2020) proposed a constrainted programming approach to solve MALBP. Their objectives are to minimize the cycle time as primary objective and to minimize the number of stations as secondary objective. Zhang et al. (2020) developed an ant colony optimization algorithm that solves MALBP in case of having space constraints.

To the best of knowledge, the only work that considers the combination of UALBP and MALBP is presented by Zakaraia et al. (2021), which the problem was sovled using stochastic local search (SLS). The proposed DE algorithm herein is more intellegent than SLS, where the proposed DE contains better priority structure for constructing feasible solutions and it contains learning procedures to ignore the worse solutions from the search space by replacing them with new random ones to increase exploration.

3. The Optimization Model

As aforementioned, this study concerns with combining UALBP and MALBP under uncertainty by having the processing times of the tasks as stochastic random variables with known means and variances and they are normally distributed. Therefore, the cycle time constraints are formulated using probabilistic constraints that are restricted by predetermined chance probability. The mathematical model can be formulated as follows:

3.1. Notations

$i=\{1,\ldots, n\}$	The set of tasks
$j=\{1,\ldots, m\}$	The set of stations
$k=\{1,\ldots,l\}$	The set of operators
ct	Cycle time
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
$IP(t_i)$	The immediate predecessors of task i
$IS(t_i)$	The immediate successors of tasks i

3.2. Decision Variables

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

$$y_j = \begin{cases} 1, \textit{ if } \sum_{i=1}^{n} x_{ijk} > 0 \\ 0, \textit{ otherwise} \end{cases}$$

The opened station decision variable

$$R_k = \begin{cases} 1, \textit{ if } \sum_{i=1}^{n} x_{ijk} > 0 \\ 0, \textit{ otherwise} \end{cases}$$

The operator assignment decision variable

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

The immediate predecessor's assignment decision variable

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

The immediate successors assignment decision variables

3.3. The Objective function

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

3.4. The Constraints

$$\sum_{j=1}^{m} x_{ijk} = 1, \ \forall i = \{1, \ldots, n\} \tag{2}$$

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

$$P\left(t_i x_{ijk} + \sum_{k=1}^{k_{max}} \sum_{g \in iP(i)} t_g x_{gjk} + \sum_{k=1}^{k_{max}} \sum_{h \in iS(i)} t_h x_{hjk} \leq ct\right) \geq \alpha, \forall j = \{1, ..., m\} \quad (4)$$

$$P\left(\sum_{i=1}^{n} t_i x_{ijk} \leq ct\right) \geq \alpha, \ \forall k = \{1, ..., l\} \quad (5)$$

$$P_i + S_i \geq 1, \ \forall i = \{1, ..., n\} \quad (6)$$

The objective function (1) seeks to minimize the number of stations and the number of operators. The set of constraints (2) is the set of assignment constraints, which ensures that each task is assigned in only one station. The set of constraints (3) shows that the maximum total processing time of the assigned tasks of any station must be less than or equal the cycle time multiplied by the maximum number of operators. The set of constraints (4) is the sequencing constraints that ensures that if any of the immediate predecessors or successors of any task are assigned on its station, then their processing time added to the task processing time must not exceed the cycle time. This set of constraints helps to avoid assigning tasks to more than one operator without considering the processing times of the predecessors and successors. The set of constraints (5) is the cycle time constraints for operators, which ensures that the total processing time of any assigned tasks to an operator must not exceed the cycle time. The sets of constraints (3), (4), and (5) are sets of chance-constraints, which are restricted by a predetermined chance probability. The set of constraints (6) is the set of predecessors and successors constraints, which ensures that each task has to be assigned either before its immediate predecessors or before immediate successors.

The mathematical model contains some chance-constraints that can be converted into deterministic in order to be solved. The processing times herein are normally distributed random variables with known means and variance. Taha (2017) shows how to overcome the chance-constraints through converting them into non-linear deterministic constraints. Therefore, the set of constraints (3), (4), and (5) can be converted into the non-linear deterministic form as shown in (7), (9), and (8) respectively.

$$\sum_{i=1}^{n} E(t_i) x_{ijk} + K_\alpha \sqrt{\sum_{i=1}^{n} Var(t_i) x_{ijk}} \leq k_{max} ct, \quad \forall j = \{1, ..., m\}, \quad (7)$$

where K_α is the standard normal value of α

$$E(t_i)x_{ijk} + \sum_{k=1}^{k_{max}} \sum_{g \in IP(i)} E(t_g)x_{gjk} + \sum_{k=1}^{k_{max}} \sum_{h \in IS(i)} E(t_h)x_{hjk} + K_\alpha \sqrt{Var(t_i)x_{ijk} + \sum_{k=1}^{k_{max}} \sum_{g \in IP(i)} Var(t_g)x_{gjk} + \sum_{k=1}^{k_{max}} \sum_{h \in IS(i)} Var(t_h)x_{hjk}} \leq ct, \forall j = \{1, ..., m\} \quad (8)$$

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

4. The Proposed DE Algorithm

DE is one of the population-based metaheuristics that consists of four phases. The first phase is the initialization, which concerns with generating the initial population of solutions. The second phase concerns with the mutation procedure. The third phase is concerned with the crossover procedure. The fourth phase is the selection procedure. All of these phases work iteratively until reaching the stopping criterion, which is herein the number of iterations.

4.1. The initialization Phase

In the initialization phase, a set of random solutions is to be generated in order to cover diversified areas of the solution space. The problem here can have random solution by generating random sequence of the tasks. Such random sequence (T) represents the priority of the tasks. So, any opened station will have the top priority tasks that satisfy the problem constraints through using the following heuristics:

Algorithm 1: The heuristic procedure

j=1, station S_j=Ø, and solution =Ø while T≠Ø do:

find the assignable tasks (AS) that ensure the problem constrains if AS≠Ø then:

Assign the highest priority (P) task in T to S_j

T=T−{P}

else:

solution = *solution* ∪ S_j

$j=j+1$ *and* $S_j=\emptyset$

return solution

DE algorithm uses vectors in its search methodology. Therefore, the random sequence of tasks can be generated using a vector that its length equal to the number of tasks and its values are randomly generated using the following equation:

$$V_i = rand\ (0,1) \tag{10}$$

Such random vector represents the position of the solution. The next step of generating the random sequence using such generated vector is to use the bubble sort algorithm as follows:

Algorithm 2: Bubble sort for creating a random sequence of tasks

T=the tasks vector arranged by index number

in ascending order

n=the number of tasks

Continue=1

i=1

while Continue=1 *do:*

Continue=0

i'=1

while $i' \leq n-i$ *do:*

if $V_{i'} \leq V_{i'+1}$ *then*

$Temporary_1 = V_{i'}$

$Temporary_2 = T_{i'}$

$V_{i'} = V_{i'+1}$

$T_{i'} = T_{i'+1}$

$V_{i'+1} = Temporary_1$

$T_{i'+1} = Temporary_2$

Continue=1

$i'=i'+1$

return T

By using equation (10), Algorithm 1, and Algorithm 2, the initial population can have a set of randomly generated candidate solutions $C(S)$. The next steps of the algorithm are to update the solutions of the population through mutation and crossover procedures and either select to keep solutions or replace them. All of these steps are to be done in iterative manner until reaching a stopping criterion, which is herein the number of iterations.

4.2. The Mutation Phase

In the mutation phase, each solution in the population is to be mutated using its positional vector. The mutation procedure uses three positions to generate new position Pos_{new}. The first position is the position of the best solution found Pos_{best}. The second position is the position of the current solution $Pos_{current}$. The third position Pos_{other} is a position of randomly selected solution from the population that isn't equal to the current solution.

$$Pos_{new} = Pos_{best} - \beta(Pos_{current} - Pos_{other}) \tag{11}$$

Algorithm 1 now is ready to be used to obtain new solution using Pos_{new}. The new solution is to be compared with the best solution found and replaces it if it is better. The parameter β is called the differential weight, which helps to define how far the new position from the three used positions.

4.3. The Crossover Phase

The crossover procedure uses both of Pos_{new} and Pos_{other}. In such process a new position is to be generated by having properties from t Pos_{new} and $Pos_{current}$. The process is controlled by a new parameter CR, which is the crossover probability. So, each value of the crossover position Pos_{cross} can be generated using the following equation:

$$Pos_{cross_i} = \begin{cases} Pos_{current_i}, & if\ rand\ (0,1) > CR \\ Pos_{new_i}, & otherwise \end{cases} \tag{12}$$

The solution of Pos_{cross} is to be generated and it will replace the best solution if it is better.

4.4. The Selection Phase

The selection phase determines the new position of the new solution in the next iteration. So, it can be replaced by Pos_{new} if it produces a better solution than the current solution, or it will be replaced by another solution that is generated randomly using a random position Pos_{random} by using equation (10).

5. Numerical Example

This section shows a numerical example to further illustrate the model and the proposed DE algorithm. The numerical example consists of 6 tasks. Its cycle time is 8 and its precedence graph is shown in Figure 1. The number of operators is 2 and the chance probability is 0.95.

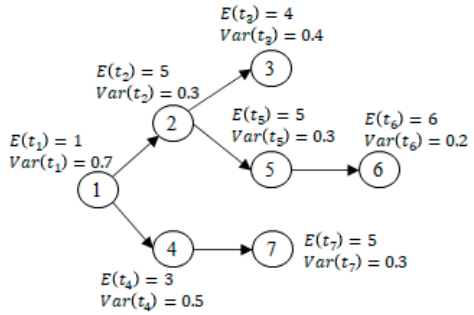

Figure 1. The precedence graph of the numerical example.

In the initialization phase, the population of solutions is to be generated using algorithm 1 and 2. So, the generation of one of these solutions can be illustrated as follows. Firstly, the number of tasks herein is 6. Therefore, one of the random vectors (RV) can be shown as follows:

Tasks	1	2	3	4	5	6
RV	0.3	0.8	0.9	0.4	0.2	0.5

After using the proposed bubble sort, algorithm 2, the arrangement of the tasks that should be used with the heuristic algorithm is as follows:

Tasks	3	2	6	4	1	5
RV	0.9	0.8	0.5	0.4	0.3	0.2

Now algorithm 1, the heuristic procedure, is ready to be used. Table 1 shows the heuristic solution using RV vector.

Table 1. The heuristic solution using RV.

Task	3	6	1	2	4	5
Processing time	4	6	1	5	3	5
Station	1	1	1	2	2	3
Operator	1	2	2	3	3	4
Completion time	4	6	7	5	8	5

The mutation phase is about to find another solution in the local space of the current solution using the linear combination found in Equation (11), where it uses the position vector of the best solution and a position vector of a random solution. If it considered that the solution found in Table 1 is the best solution, then its position vector, which is the sorted RV vector (SRV), is to be used along with another position vector of a random solution to find a neighbor of the current solution. For illustration, VC represents the position vector of the current position, VCR represents the position vector of a random solution, and VCN represents the position vector of the neighbor solution. Table 2 shows the VCN after using β=0.3.

Table 2. Generating neighbour using mutation phase.

Tasks	1	2	3	4	5	6
VC	0.9	0.6	0.5	0.4	0.2	0.1
SRV	0.9	0.8	0.5	0.4	0.3	0.2
VCR	0.9	0.8	0.7	0.3	0.2	0.1
VCN	0.9	0.9	0.6	0.35	0.3	0.2

The crossover phase is about to generated new position vector using the position vector of the current solution and the position of the random solution, where this process selects characteristics from both solutions. Table 3 shows the crossover process, where the highlighted numbers show the selected characteristics from each solution.

Table 3. Generating new solution using crossover process.

Tasks	1	2	3	4	5	6
VC	0.9	0.6	0.5	0.4	0.2	0.1
VCR	0.9	0.8	0.7	0.3	0.2	0.1
VCN	0.9	0.6	0.5	0.3	0.2	0.1

6. Experimental Design

The proposed algorithm is developed using python programming in PC that has 2.93 GHz core2duo CPU and 4 GB rams. It has five parameters, which are the population size Pop_{size}, the number of iterations $Maxit$, the minimum values of the differential weight β_{min}, the maximum value of the differential weight β_{max}, and the crossover probability CP. Each parameter has four levels shown in Table 4. The number of experiments required to make the full factorial design is 4^5=1024 experiments. Such number of experiments can be radically reduced using the Taguchi method by having L_{16} orthogonal array, which only have 16 experiments. The corresponding L_{16} orthogonal array for the current experimental design is in shown in Table 5.

Table 4. The parameter levels of the experimental design.

Pop_{size}	$Maxit$	β_{min}	β_{max}	CP
25	25	-1	0	0.1
50	50	-0.5	0.25	0.2
75	75	0	0.5	0.3
100	100	0.5	1	0.4

Table 5. The required orthogonal array for the experimental design.

Trail	Pop_{size}	*Maxit*	β_{min}	β_{max}	*CP*
1	25	25	-1	0	0.1
2	25	50	-0.5	0.25	0.2
3	25	75	0	0.5	0.3
4	25	100	0.5	1	0.4
5	50	25	-0.5	0.5	0.4
6	50	50	-1	1	0.3
7	50	75	0.5	0	0.2
8	50	100	0	0.25	0.1
9	75	25	0	1	0.2
10	75	50	0.5	0.5	0.1
11	75	75	-1	0.25	0.4
12	75	100	-0.5	0	0.3
13	100	25	0.5	0.25	0.3
14	100	50	0	0	0.4
15	100	75	-0.5	1	0.1
16	100	100	-1	0.5	0.2

The selected problems for design of experiments are taken from well-known benchmarks can be found in https://assembly-line-balancing.de/salbp/. The problems included in such benchmarks are deterministic and need to be adapted to fit the mathematical model of this paper. Therefore, the processing times of tasks in the selected problems must have expected values and variances. In order to adapt the problems, the expected values of the processing times are considered the same as the values of the original processing times and the variances are calculated by subtracting each processing time from the expected value and divide the output by 1000. Table 6 shows the selected problems for the experimental design.

Table 6. The selected problems for experimental design.

Serial	Problem	Cycle time	Number of tasks
1	JACKSON	7	11
2	JACKSON	9	11
3	MITCHELL	14	21
4	MITCHELL	15	21
5	HESKIA	138	28
6	HESKIA	205	28
7	SAWYER30	25	30
8	SAWYER30	27	30
9	ARC83	5048	83
10	ARC83	5853	83
11	ARC111	5755	111
12	ARC111	8847	111

The response value for the experimental design includes the value of the objective function and the CPU time where that leads to better solutions with a little time consumption. Equation (13) shows the required response value for each trail in the experimental design.

$$Response = \sum_{j=1}^{m} y_j - \frac{1}{\sum_{k=1}^{l} R_k} - \frac{1}{CPU\,time} \qquad (13)$$

The selected problems are different and each has different solution and response value. Therefore, the response values are normalized as shown in Table 7.

The analysis of variance for the parameter levels is done in order to study main effects. Table 8 shows the F-value and P-value for each parameter.

Table 7. The normalized values for the responses for each selected problem.

Trail	1	2	3	4	5	6	7	8	9	10	11	12
1	0.062	0.063	0.063	0.062	0.062	0.069	0.061	0.062	0.061	0.062	0.063	0.062
2	0.062	0.063	0.063	0.063	0.060	0.061	0.062	0.059	0.061	0.058	0.063	0.064
3	0.062	0.063	0.062	0.063	0.063	0.057	0.058	0.060	0.063	0.063	0.063	0.063
4	0.062	0.062	0.062	0.062	0.061	0.063	0.062	0.067	0.064	0.063	0.063	0.064
5	0.062	0.063	0.062	0.063	0.065	0.073	0.060	0.061	0.064	0.064	0.063	0.059
6	0.062	0.063	0.062	0.062	0.061	0.056	0.063	0.061	0.067	0.058	0.063	0.064
7	0.063	0.063	0.062	0.062	0.064	0.069	0.063	0.067	0.067	0.064	0.063	0.063
8	0.063	0.062	0.062	0.062	0.064	0.061	0.058	0.067	0.061	0.063	0.063	0.061
9	0.062	0.063	0.062	0.062	0.061	0.069	0.067	0.061	0.061	0.058	0.063	0.063
10	0.062	0.062	0.063	0.063	0.062	0.057	0.061	0.059	0.061	0.064	0.063	0.062
11	0.063	0.063	0.062	0.063	0.065	0.060	0.063	0.065	0.061	0.065	0.059	0.063
12	0.062	0.063	0.062	0.063	0.066	0.055	0.063	0.066	0.061	0.065	0.063	0.062
13	0.062	0.063	0.063	0.063	0.065	0.058	0.063	0.059	0.065	0.063	0.063	0.062
14	0.063	0.063	0.062	0.062	0.060	0.060	0.063	0.059	0.060	0.065	0.059	0.061
15	0.062	0.063	0.062	0.062	0.061	0.072	0.069	0.066	0.061	0.064	0.063	0.063
16	0.062	0.063	0.063	0.063	0.061	0.062	0.061	0.060	0.061	0.059	0.061	0.065

Table 8. The analysis of variance for each parameter.

Parameter	F-value	P-value
Pop_{size}	0.59	0.62
Maxit	3.54	0.016
β_{min}	1.42	0.24
β_{max}	1.60	0.19
CP	0.45	0.71

The null hypothesis is accepted in all parameters except in the number of iterations. Therefore, the Tukey's honest significant difference test is applied to show which parameter levels differ. Figure 2 shows that the worst parameter level for the number of iterations is 75 iterations and there is no significant difference between the remaining levels.

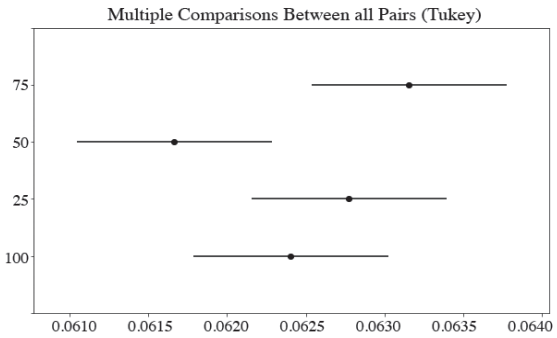

Figure 2. Tukey's interval plot for the number of iterations parameter.

7. Computational Results

This section shows the results of applying the proposed DE algorithm on 71 adapted problems for the same benchmarks found in the experimental design section. Table 9 shows the computational results.

Table 9. The computational results.

Problem	Problem size	Cycle time	Result	CPU time
MERTENS	7	6	3.83	0.001
MERTENS	7	7	2.83	0.001
MERTENS	7	8	2.83	0.001
MERTENS	7	10	2.75	0.001
MERTENS	7	15	1.5	0.001
MERTENS	7	18	0.5	0.005
BOWMAN8	8	17	4.83	0.001
BOWMAN8	8	20	3.8	0.003
BOWMAN8	8	21	3.8	0.005
BOWMAN8	8	24	3.8	0.001
BOWMAN8	8	28	2.8	0.003
BOWMAN8	8	31	1.67	0.001
JAESCHKE	9	6	5.88	0.001
JAESCHKE	9	7	5.86	0.001
JAESCHKE	9	8	5.86	0.002
JAESCHKE	9	10	3.8	0
JAESCHKE	9	18	2.67	0
JACKSON	11	7	5.88	0.029
JACKSON	11	9	4.86	0.001
JACKSON	11	10	3.83	0.003
JACKSON	11	13	2.75	0.005
JACKSON	11	14	2.75	0.001
JACKSON	11	21	1.67	0.001
MANSOOR	11	45	2.8	0.019
MANSOOR	11	54	2.75	0.001
MANSOOR	11	63	1.67	0.026
MANSOOR	11	72	1.67	0.001
MANSOOR	11	81	1.67	0.001
MITCHELL	21	14	7.9	0.002
MITCHELL	21	15	6.9	0.004
MITCHELL	21	21	3.86	0.023
MITCHELL	21	26	2.8	2.247
MITCHELL	21	35	2.75	0.003
MITCHELL	21	39	1.67	2.439
HESKIA	28	138	4.88	0.192
HESKIA	28	205	2.83	0.833
HESKIA	28	216	2.8	0.086
HESKIA	28	256	2.8	0.009
HESKIA	28	324	1.75	0.013
HESKIA	28	342	1.75	0.014
SAWYER30	30	25	7.94	6.457
SAWYER30	30	27	7.93	0.51
SAWYER30	30	30	7.92	0.132
SAWYER30	30	36	5.9	0.016
SAWYER30	30	41	4.89	8.426
SAWYER30	30	54	3.86	0.027
SAWYER30	30	75	2.8	0.008
KILBRID	45	57	5.9	1.401
KILBRID	45	79	3.88	0.016
KILBRID	45	92	3.86	0.01
KILBRID	45	110	2.83	0.277
KILBRID	45	138	2.8	0.009
KILBRID	45	184	1.67	15.791
TONGE70	70	176	11.96	0.447
TONGE70	70	364	5.91	0.092
TONGE70	70	410	4.89	1.524
TONGE70	70	468	3.88	4.191
TONGE70	70	527	3.86	0.018
ARC83	83	5048	8.94	18.726
ARC83	83	5853	7.93	3.26
ARC83	83	6842	6.92	32.885
ARC83	83	7571	5.91	9.666
ARC83	83	8412	5.9	1.057
ARC83	83	8998	4.89	7.431
ARC83	83	10816	3.88	1.209
ARC111	111	5755	15.97	31.141
ARC111	111	8847	9.95	0.43
ARC111	111	10027	8.94	13.533
ARC111	111	10743	7.93	88.503
ARC111	111	11378	7.93	3.89
ARC111	111	17067	4.89	8.653

8. Conclusion

The problem handled in this paper considers a combination between UALBP and MALBP under uncertainty. Such combination leads to minimize the line length through having more than one operator in any station and utilizing the flexibility of the task's assignment in U-shaped lines. The processing times of the tasks differ from operator to another, where that leads to uncertain values of them. Thus, the processing times of the tasks are represented as random variables with known means and variances. Therefore, the cycle time constraints of the mathematical model for such combined problem are represented as chance-constraints. The proposed approach for solving the problem is DE algorithm. The algorithm parameters are optimized and 71 adapted problems have been solved as a computational result. The future points of research may include the following:

- Formulating the same problem with another type of uncertainty such as fuzzy and rough programming.
- Including space constraints.
- Including worker assignment.
- Proposing other approaches for solving the same problem.

References

Abidin Çil, Zeynel, & Damla Kizilay. 2020. Constraint Programming Model for Multi-Manned Assembly Line Balancing Problem. *Computers and Operations Research, 124*, 105069. https://doi.org/10.1016/j.cor.2020.105069

Ajenblit, Debora A., & Roger L. Wainwright. 1998. Applying Genetic Algorithms to the U-Shaped Assembly Line Balancing Problem. *Proceedings of the IEEE Conference on Evolutionary Computation, ICEC,* 96–101. https://doi.org/10.1109/icec.1998.699329

Fathi, Masood, María Jesús Álvarez, & Victoria Rodríguez. 2016. A New Heuristic-Based Bi-Objective Simulated Annealing Method for U-Shaped Assembly Line Balancing. *European Journal of Industrial Engineering, 10*(2), 145–169. https://doi.org/10.1504/EJIE.2016.075849.

Fattahi, Parviz, Abdolreza Roshani, & Abdolhassan Roshani. 2011a. A Mathematical Model and Ant Colony Algorithm for Multi-Manned Assembly Line Balancing Problem. *International Journal of Advanced Manufacturing Technology, 53*(1–4), 363–378. https://doi.org/10.1007/s00170-010-2832-y

Gökçen, Hadi, Kürşad Ağpak, & Recep Benzer. 2006. Balancing of Parallel Assembly Lines. *International Journal of Production Economics,* 103(2), 600–609. https://doi.org/10.1016/j.ijpe.2005.12.001

Hamzadayi, Alper, & Gokalp Yildiz. 2012. A Genetic Algorithm Based Approach for Simultaneously Balancing and Sequencing of Mixed-Model U-Lines with Parallel Workstations and Zoning Constraints. *Computers and Industrial Engineering, 62*(1), 206–215. https://doi.org/10.1016/j.cie.2011.09.008

Hamzadayi, Alper, & Gokalp Yildiz. 2013. A Simulated Annealing Algorithm Based Approach for Balancing and Sequencing of Mixed-Model U-Lines. *Computers and Industrial Engineering, 66*(4), 1070–1084. https://doi.org/10.1016/j.cie.2013.08.008

Jayaswal, Sachin, & Prashant Agarwal. 2014. Balancing U-Shaped Assembly Lines with Resource Dependent Task Times: A Simulated Annealing Approach. *Journal of Manufacturing Systems, 33*(4), 522–534. https://doi.org/10.1016/j.jmsy.2014.05.002

Kara, Yakup, Cemal Özgüven, Neşe Yalçin, & Yakup Atasagun. 2011. Balancing Straight and U-Shaped Assembly Lines with Resource Dependent Task Times. *International Journal of Production Research, 49*(21), 6387–6405. https://doi.org/10.1080/00207543.2010.535039

Kellegöz, Talip. 2017. Assembly Line Balancing Problems with Multi-Manned Stations: A New Mathematical Formulation and Gantt Based Heuristic Method. *Annals of Operations Research, 253*(1), 377–404. https://doi.org/10.1007/s10479-016-2156-x

Kellegöz, Talip, & Bilal Toklu. 2015. A Priority Rule-Based Constructive Heuristic and an Improvement Method for Balancing Assembly Lines with Parallel Multi-Manned Workstations. *International Journal of Production Research, 53*(3), 736–756. https://doi.org/10.1080/00207543.2014.920548

Kucukkoc, Ibrahim, & David Z. Zhang. 2015. *Balancing of Parallel U-Shaped Assembly Lines*. Vol. 64. Virginia Tech.

Li, Ming, Qiuhua Tang, Qiaoxian Zheng, Xuhui Xia, & C. A. Floudas. 2017. Rules-Based Heuristic Approach for the U-Shaped Assembly Line Balancing Problem. *Applied Mathematical Modelling, 48*(2017), 423–439. https://doi.org/10.1016/j.apm.2016.12.031

Li, Zixiang, Mukund Nilakantan Janardhanan, & Humyun Fuad Rahman. 2021. Enhanced Beam Search Heuristic for U-Shaped Assembly Line Balancing Problems. *Engineering Optimization, 53*(4), 594–608. https://doi.org/10.1080/0305215X.2020.1741569

Michels, Adalberto Sato, Tiago Cantos Lopes, Celso Gustavo Stall Sikora, & Leandro Magatão. 2018. With Practical Extensions The Robotic Assembly Line Design (RALD) Problem: Model and Case Studies with Practical Extensions. *Computers & Industrial Engineering, 120,* 320–333. https://doi.org/10.1016/j.cie.2018.04.010

Michels, Adalberto Sato, Thiago Cantos Lopes, Celso Gustavo Stall Sikora, & Leandro Magatão. 2019. A Benders' Decomposition Algorithm with Combinatorial Cuts for the Multi-Manned Assembly Line Balancing Problem. *European Journal of Operational Research, 278*(3), 796–808. https://doi.org/10.1016/j.ejor.2019.05.001

Miltenburg, G.J., & J. Wijngaard. 1994. U-Line Line Balancing Problem. *Management Science, 40*(10), 1378–1388. https://doi.org/10.1287/mnsc.40.10.1378

Nourmohammadi, Amir, Masood Fathi, Mostafa Zandieh, & Morteza Ghobakhloo. 2019. A Water-Flow like Algorithm for Solving U-Shaped Assembly Line Balancing Problems. *IEEE Access, 7*, 129824–129833. https://doi.org/10.1109/ACCESS.2019.2939724

Oksuz, Mehmet Kursat, Kadir Buyukozkan, & Sule Itir Satoglu. 2017. U-Shaped Assembly Line Worker Assignment and Balancing Problem: A Mathematical Model and Two Meta-Heuristics. *Computers and Industrial Engineering, 112*, 246–263. https://doi.org/10.1016/j.cie.2017.08.030

Rabbani, Masoud, Seyed Mahmood Kazemi, & Neda Manavizadeh. 2012. Mixed Model U-Line Balancing Type-1 Problem: A New Approach. *Journal of Manufacturing Systems, 31*(2), 131–138. https://doi.org/10.1016/j.jmsy.2012.02.002

Sabuncuoglu, Ihsan, Erdal Erel, & Arda Alp. 2009. Ant Colony Optimization for the Single Model U-Type Assembly Line Balancing Problem. *International Journal of Production Economics, 120*(2), 287–300. https://doi.org/10.1016/j.ijpe.2008.11.017

Scholl, A., & R. Klein. 1999. ULINO: Optimally Balancing U-Shaped JIT Assembly Lines. *International Journal of Production Research, 37*(4), 721–736. https://doi.org/10.1080/002075499191481

Sresracoo, Poontana, Nuchsara Kriengkorakot, Preecha Kriengkorakot, & Krit Chantarasamai. 2018. U-Shaped Assembly Line Balancing by Using Differential Evolution Algorithm. *Mathematical and Computational Applications, 23*(4), 79. https://doi.org/10.3390/mca23040079

Taha, Hamdy. 2017. *Operations Research an Introduction*. 10th ed. edited by R. Horton, Marcia; Partridge, Julian; Stark, Holly; Brands, Amanda; Agarwal, Aditee; Raheja. Harlow: Pearson Education Limited.

Yilmaz, Ö.F., Ö.F. Demirel, S. Zaim, & S. Sevim. 2020. Assembly Line Balancing by Using Axiomatic Design Principles: An Application from Cooler Manufacturing Industry. *International Journal of Production Management and Engineering, 8*(1), 31–43. https://doi.org/10.4995/IJPME.2020.11953

Yılmaz, Faruk. 2020a. Robust Optimization for U-Shaped Assembly Line Worker Assignment and Balancing Problem with Uncertain Task Times¨Omer Times¨ Times¨Omer. *CRORR, 11*(2), 229–239. https://doi.org/10.17535/crorr.2020.0018

Yılmaz, Ömer Faruk. 2020b. An Integrated Bi-Objective U-Shaped Assembly Line Balancing and Parts Feeding Problem: Optimization Model and Exact Solution Method. *Annals of Mathematics and Artificial Intelligence,* 1–18. https://doi.org/10.1007/s10472-020-09718-y

Zakaraia, Mohammad, Hegazy Zaher, & Naglaa Ragaa. 2021. Stochastic Local Search for Solving Chance-Constrained Multi-Manned U-Shaped Assembly Line Balancing Problem with Time and Space Constraints. *Journal of University of Shanghai for Science and Technology, 23*(04), 278–295. https://doi.org/10.51201/JUSST/21/04242

Zhang, Beikun, & Liyun Xu. 2020. An Improved Flower Pollination Algorithm for Solving a Type-II U-Shaped Assembly Line Balancing Problem with Energy Consideration. *Assembly Automation, 40*(6), 847–856. https://doi.org/10.1108/AA-07-2019-0144

Zhang, Zikai, Qiuhua Tang, & Manuel Chica. 2020. Multi-Manned Assembly Line Balancing with Time and Space Constraints: A MILP Model and Memetic Ant Colony System. *Computers and Industrial Engineering, 150*, 106862. https://doi.org/10.1016/j.cie.2020.106862

