Received: April 11, 2021. Revised: May 25, 2021.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Modified Differential Evolution Algorithm for U-Shaped Assembly Line Balancing Type 2

Krit Chantarasamai[1] **On-Uma Lasunon[1]***

[1]*Manufacturing and Materials Research Unit, Faculty of Engineering, Mahasarakham University, Thailand*
* Corresponding author's Email: onuma.l@msu.ac.th

Abstract: We describe a Differential Evolution (DE) algorithm for solving U-shaped Assembly Line Balancing Problems – Type 2 (UALBP-2). The minimum cycle time in a just-in-time production line for producing a single product with a certain number of workstations was investigated by developing solution methods and testing on 15 problem sets (101 instances). The problems were divided into 10 medium-scale problem sets (50 instances) and 5 large-scale problem sets (51 instances). When comparing the results with those obtained from the rule-based heuristic; three rules and two rules, the DE algorithm could generate 14 better solutions (28%) in the medium-scale problems, and 3 better solutions (about 6%) in the large-scale problems. In terms of computational time, DE algorithm was faster than rule-based heuristic methods in medium-scale problems. In large problems, our DE algorithm used computational times that were shorter by 73% (for two rules) and 98% (for three rules).

Keywords: Differential evolution algorithm, Just-in-time, U-shaped assembly line balancing problems.

1. Introduction

Assembly line balancing is an important problem for many industries, especially industries that are constantly adapting assembly lines for new products to respond to customer needs for delivery, raw materials, products, and services in the shortest possible time, or "just-in-time" (JIT) manufacturing. In on-time production and delivery of products, cycle time is a key factor in balancing an assembly line in a JIT system. The type of assembly line balancing is another factor affecting the efficiency of line balancing.

Assembly line balancing is a form of production planning used to assign appropriate tasks to each workstation in order to reduce cycle time or the number of workstations, increase flexibility in process flow, and eliminate delays or bottlenecks during production. There are two types of assembly line balancing problems: simple assembly line balancing problem (SALBP) and general assembly line balancing problem (GALBP) [1, 2], as shown in Fig. 1. SALBP can be classified into four types based on the balancing objective, i.e., SALBP-1 to minimize the number of workstations (m), SALBP-2 to minimize the cycle time (c), SALBP-E to find the optimal efficiency of the assembly line, and SALBP-F to balance the operations at each workstation. GALBP can be classified into three or more types, i.e., mixed model assembly line balancing problem (MALBP), U-shaped assembly line balancing problem (UALBP), and some others. The MALBP models the production of mixed products and multiple models of them. In the UALBP, a worker can work on both sides of a U-shaped assembly line: it is also divided into three types: UALBP-1 to minimize the number of workstations (m), UALBP-2 to minimize the cycle time (c), and UALBP-E to find the optimal efficiency of the line. The third type of GALBP is an assembly line balancing problem with a wide range of applications, which can be considered for many conditions.

In many industries, straight SALBP lines (see Fig. 2 (a)) and U-shaped GALBP lines (see Fig. 2 (b)) are normally used to minimize the cycle time. However, U-shaped line balancing offers more advantages than straight line balancing: it has higher

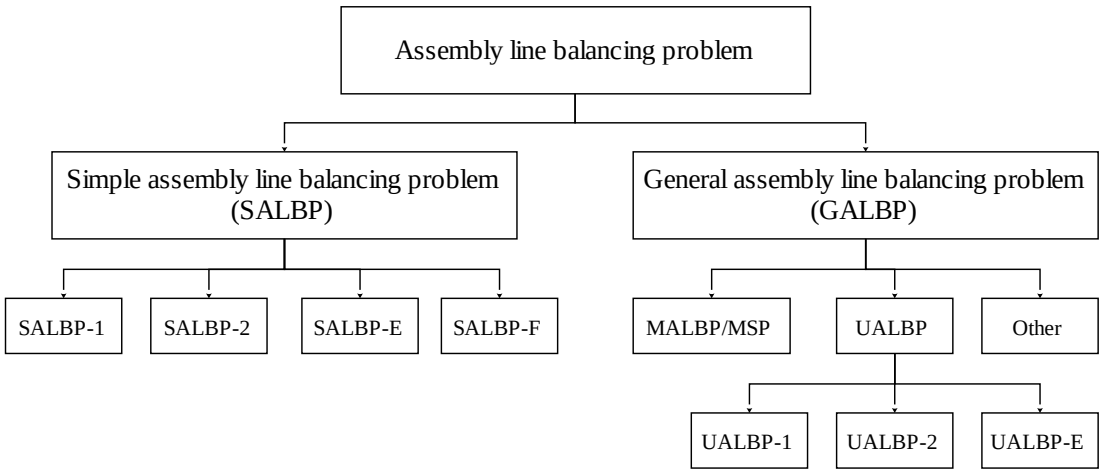
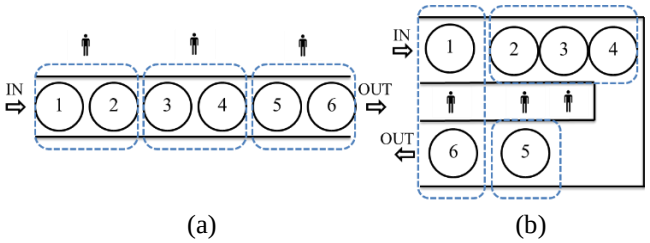


Figure. 1 Taxonomy of assembly line balancing problems [1]



(a) (b)

Figure. 2 Assembly line layouts: (a) Straight assembly line and (b) U-shaped assembly line

efficiency and flexibility, fewer workstations, and lower cycle time [3-5]. Therefore, we focused on solving problems with U-shaped assembly line balancing. In the following, we present review papers that deal with UALBP.

U-shaped assembly line balancing problem (UALBP) was first described in 1994 by Miltenburg and Wijngaard [6], who solved the 7-11 tasks problem with dynamic programming (DP) and the 21-111 tasks problem with the ranked positional weight heuristic (RPW). Thereafter, various approaches were developed and applied to solve the UALBP. For example, Urban [7] used integer linear programming for the UALBP formulation and solved it using CPLEX software. Scholl and Klein [8] solved the solution of the UALBP using the ULINO method. Hwang, Katayama and Gen [9] presented a multi-objective genetic algorithm (GA) (line efficiency and workload variation) for solving the 21-111 tasks problem and compared with integer programming and RPW. Baykasoğlu and Dereli [10] applied ant colony optimization (ACO) for the UALBP and solved the 8-297 tasks problem using Visual Basic software. Fathi, Alvarez and Rodríguez [11] presented the critical task method (CTM) for solving the UALBP and compared it with 12 other heuristics. Kriengkorakot [12] applied heuristics (maximum task time, minimum task time, maximum ranked positional weight and Greedy randomized) to improve assembly line balancing in a garment factory. Sresracoo, Kriengkorakot and Chantarasamai [13] applied a differential evolution algorithm (DE) to the UALBP-1 and solved the 21-297 tasks problem using NetBeans software. Zhang, Tang, Han and Li [14] proposed an enhanced migrating birds optimization algorithm (EMBO) for UALBP with worker assignment. Li, Hu, Tang and Kucukkoc [15] developed an algorithm based on the branch and bound remember algorithm to solve UALBP-1 with uncertain task time. Varnasilpin and Masuchun [16] presented the uncertain problem of three U-shaped lines defining time as an interval for the UALBP and solved it using MATLAB R2014a software.

In reviewing U-shaped assembly line balancing problem, it was found that many researchers have specifically solved UALBP-1, while UALBP-2, which is important for JIT, is still limited. Jonnalagedda and Dabade [4] in 2014 used a genetic algorithm (GA) to solve small and medium sized problems of the UALBP-2 and found that the algorithm can solve the problem very well for a small number of workstations. In 2017, Li et al. [5] applied rule-based heuristics (two rules and three rules) to solve medium and large problems of UALBP-2 and found that the algorithm performs well in terms of efficiency and computational speed compared to integer programming, but the uncertainty of its computational accuracy limits its application.

Assembly line balancing type 2 (ALBP-2) is important for JIT as it can minimize the cycle time for a given number of workstations. ALBP-2 was first reported by Klein and Scholl in 1996 to solve the simple assembly line balancing problem - type 2 (SALBP-2) using branch-and-bound methods [17]. After that, many researchers have proposed heuristic

methods to solve assembly line problems, e.g. Liu, Ong and Huang [18, 19] presented two bi-directional heuristics for SALBP-2 to minimize the cycle time and found that the algorithm was efficient in minimizing both the cycle time and the mean absolute deviation. Seyed-Alagheband, Fatemi Ghomi and Zandieh [20] applied the simulated annealing algorithms (SA) to solve sequence-dependent setup time problems with the objective of minimizing the cycle time for a given number of workstations and found that this algorithm was effective in terms of computation time and optimal solutions. Jirasirilerd et al. [2] presented a variable neighborhood strategy adaptive search (VaNSAS) to minimize the cycle time for the SALBP-2 problem in the garment industry considering the number and type of machines used at each workstation and found that the algorithm provides a better solution and much less computation time compared to the program LINGO.

After reviewing UALBP and ALBP-2, we found that the heuristic and metaheuristic methods are effective in solving problems. Many researchers have been engaged in solving problems for different purposes by using different methods like SA, CTM, GA, ACO and DE. However, the differential evolution algorithm (DE), which is considered as the optimal metaheuristic method because of its speed and accuracy [21, 22], has attracted our attention.

The DE algorithm was first described by Storn and Price [23] in 1997. It has been used to solve a wide variety of problems and objectives in, for example, assignment problem to minimize cost [22], transportation problem to maximize profit and minimize cost of transport [24-26], location routing problem to minimize the fuel usage [27], simple assembly line balancing problem-type 1 (SALBP-1) to minimize number of workstation [28, 29] and U-shaped assembly line balancing problem-type 1 (UALBP-1) to minimize number of workstation [13]. Ramadas, Abraham and Kumar [30] proposed a new revised mutation strategy in DE algorithm to improve the optimal solution. Recently, Srichok et al. [31] combined the DE algorithm with response surface method (RSM) to find the optimal parameters for friction stir welding. For assembly line balancing problems, the results showed that the DE algorithm is more efficient and requires less computation time compared to GA and Tabu Search (TS) in SALBP-1 [28] and with ACO in UALBP-1 [13]. However, the DE algorithm has not yet been used to solve the U-shaped assembly line balancing problem - type 2 (UALBP-2).

Since this study aimed to improve the efficiency of assembly line balancing for JIT systems, we therefore developed a heuristic method using the DE algorithm to solve the UALBP-2 to minimize the cycle time for a given number of workstations. The UALBP-2 was solved by applying the Java program in the NetBeans software. To evaluate the efficiency of our modified DE algorithm, our results should be compared with other efficient methods. However, there were few reports on UALBP-2 [4, 5] due to its complexity and difficult to solve problem, especially when the number of tasks increases. Therefore, we decided to use the same data and compare the results with Li et al. [5] who used the rule-based heuristic methods to solve medium problems with 21-58 tasks and large problems with 70-297 tasks.

The structure of this paper is as follows: Section 2 describes the overall algorithm of our modified DE. Section 3 presents the result and discussion. The advantages of our modified DE algorithm over the rule-based heuristic methods in terms of minimized cycle time and computation speed are presented. Section 4 draws the conclusions and presents the future direction of our work.

2. Our modified DE algorithm

Our modified DE algorithm has four steps: (1) generation of the initial target vector, (2) mutation, (3) recombination, and (4) selection. In the first step, an initial target vector for each task is generated using a random number between 0 and 1. These target vectors are used to choose the assignments of tasks to workstations. Number of workstations, *m*, and the number of possible initial solutions, *NP*, are set. In the second step, the DE algorithm uses mutation by 1) finding the difference of the randomized vectors from the population, 2) multiplying this difference by the scaling factor, *F*, and 3) adding to another randomized vector. In the following, we refer to the population to be mutated as the "target vector" and the adjusted or mutated one as the "mutant vector". In the recombination step, the target and mutant vectors are combined, using a settable factor, the crossover rate, *CR*, to make a decision. The recombined vector is referred to as the "trial vector". The trial vector was evaluated using the fitness function. After that, the fitness for the trial vector were compared with the target vector to find the best vector to be used as the target vector for the next evaluation. The DE algorithm loop was repeated for a specified number of rounds, *R*, and then the final result was displayed. Steps of our DE algorithm are shown in Fig. 3 More details will be described and illustrated in the next sections. Application of our DE algorithm to a UALBP-2 problem was illustrated using a small example taken from Jackson [32]: the

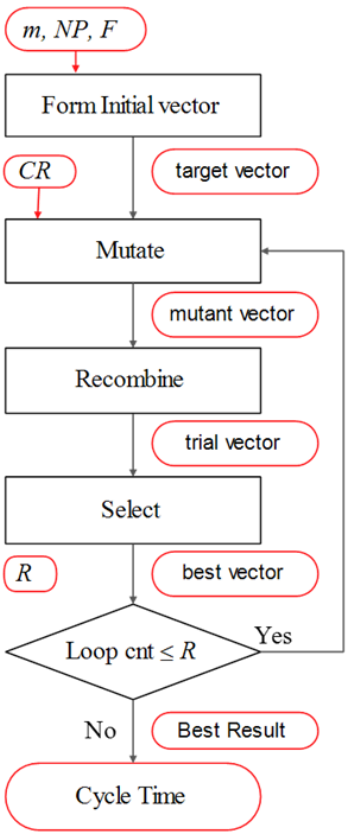


Figure. 3 Data flow for our differential evolution (DE) algorithm in UALBP-2

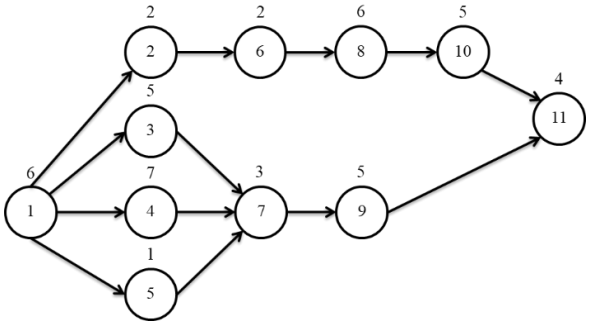


Figure. 4 Precedence diagram for the sample problem, taken from Jackson [32]

precedence diagram for its 11 tasks is shown in Fig. 4

In Fig. 4, numbers in the circles label the tasks and numbers next to the circles show the time of each task. In balancing a UALBP-2, these variables must be set: *m* = number of workstations, *R* = number of rounds, *NP* = number of possible initial solutions, *F* = scaling factor and *CR* = crossover rate. For this sample, we set: *m*=4 workstations, *R*=1 round, *NP*=5, *F*=0.8 and *CR*=0.8. Details of the calculation follow in sections 2.1-2.4.

2.1 Generate the initial target vector

In this sample task, the number of possible initial solutions, *NP*=5, and five sets of target vectors were generated - see Table 1. The initial target vectors, V_i where $i = (1 \dots NP)$, are sets of D random numbers: $V_i = \{r_1, \dots, r_D\}$ where $r_j \in [1,0] \mid j = 1, D$.

Each r_j was assigned to one of the 11 tasks. These target vectors were used to make decisions for assigning tasks to workstations. The task with the highest target vector was determined first. If it had no conflict with the precedence diagram, it was assigned to the workstation, else another task with a lower target vector was considered. Note that U-shape assembly line balancing can be determined both forward and backward directions in the precedence diagram. For example, task 11 of Vector 1 has the highest target vector, 0.88, so it is a candidate for the first task, and, having no conflicts in Fig. 4, it was assigned to the first workstation, see Table 1 and Fig. 4. The next tasks considered was task 7, with the target vector of 0.85, but task 7 may not be assigned before task 9 for the backward direction, or before tasks 3, 4, and 5 for the forward direction, see Fig. 4. Therefore, task 5 was evaluated next. This process was run until the appropriate task was assigned to the workstation. Other condition governing assigning tasks is the cycle time for each workstation, which can be calculated by dividing the total task times with the number of workstations. Each workstation must have task times lower than or equal to the cycle time. In this case, we used a cycle time of 12; therefore, tasks 11, 1, and 5 with total times of 11 were assigned to the first workstation, as shown in Table 2. The next workstations were determined using the same criteria. Possible initial solutions are shown in Table 2

2.2 Mutation process

Mutation adjusts values in vectors (Dimension: D) to obtain new solutions that differ from the initial population. In the following, two subscripts are added to each vector, $V_{i,G+1}$, where $i \in \{1, NP\}$ is the vector index and $G \in \{1, R\}$ is a generation number. Thus, the mutant vector, $V_{i,G+1}$, was calculated using the following Eq. (1) [28].

$$V_{i,G+1} = X_{best,G} + F\left(X_{r1,G} - X_{r2,G}\right) + F\left(X_{r3,G} - X_{r4,G}\right) \quad (1)$$

where $X_{best,G}$ is the best vector or the vector assigned to workstations with the minimum cycle time, $X_{r1,G}, X_{r2,G}, X_{r3,G}, X_{r4,G}$ are random vectors from population, *NP*, and F = scaling factor, a real constant between 0 and 2.

Table 1. Target vector of initial solutions

Task	1	2	3	4	5	6	7	8	9	10	11
Vector 1	0.58	0.43	0.71	0.25	0.78	0.77	0.85	0.16	0.25	0.44	0.88
Vector 2	0.25	0.54	0.44	0.80	0.42	0.70	0.60	0.12	0.75	0.32	0.44
Vector 3	0.77	0.60	0.66	0.55	0.80	0.58	0.42	0.10	0.23	0.23	0.50
Vector 4	0.59	0.44	0.72	0.26	0.79	0.78	0.86	0.17	0.26	0.45	0.89
Vector 5	0.24	0.53	0.43	0.79	0.41	0.69	0.59	0.11	0.74	0.31	0.43

Table 2. Results for initial solutions

Initial solution	Dimension	1	2	3	4	5	6	7	8	9	10	11
Vector 1	Workstation	1			2			3			4	
	Task	11	1	5	3	10	2	6	4	7	9	8
	Target Vector	0.88	0.58	0.78	0.71	0.44	0.43	0.77	0.25	0.85	0.25	0.16
Vector 2	Workstation	1			2			3			4	
	Task	11	9	7	4	3	5	10	1	2	6	8
	Target Vector	0.44	0.75	0.60	0.80	0.44	0.42	0.32	0.25	0.54	0.70	0.12
Vector 3	Workstation	1			2			3			4	
	Task	1	5	3	2	6	4	11	7	10	9	8
	Target Vector	0.77	0.80	0.66	0.60	0.58	0.55	0.50	0.42	0.23	0.23	0.10
Vector 4	Workstation	1			2			3			4	
	Task	11	1	5	3	10	2	6	4	7	9	8
	Target Vector	0.89	0.59	0.79	0.72	0.45	0.44	0.78	0.26	0.86	0.26	0.17
Vector 5	Workstation	1			2			3			4	
	Task	11	9	7	4	3	5	10	1	2	6	8
	Target Vector	0.43	0.74	0.59	0.79	0.43	0.41	0.31	0.24	0.53	0.69	0.11

From Table 2, where vector 1 was the best vector ($X_{best,G}$) and *F*=0.8, when the target vector 1 was mutated, 4 vectors (r_1, r_2, r_3, r_4) were randomized from the target vector of the initial population, *NP*, vectors 3, 2, 5 and 4 were obtained. Then, the mutant vector, $V_{i,G+1}$, was calculated by placing the obtained randomized vector in Eq. (2), the obtained mutant vector 1 was $V_{i,G+1} = 0.88 + 0.8x(0.77 - 0.44) + 0.8x(0.43 - 0.89) = 0.78$ When the calculation was complete, the mutant vectors were shown in Table 3.

2.3 Recombination step

Recombination enhances the diversity of approaches to find solutions. In this step, the trial vector will be generated by using Exponential Crossover 1 Position [28] in Eq. (3), starting with a set of random numbers between 0 and 1 in every position, r_j. Then search for the first position, j, so that $r_j < CR$. The trial vector after this position will be equal to the target vector; else the trial vector will be equal to the mutant vector.

$$U_{ji,G+1} = \begin{cases} V_{ji,G+1} \; where \; n \in \{1 \dots r\} \; and \\ \quad the \; first \; r \; that \; is \; found \; (r_j < CR) \\ X_{ji,G} \; Other \end{cases} \quad (2)$$

where $U_{ji,G+1}$ = trial vector, $V_{ji,G+1}$ = mutant vector, $X_{ji,G}$ = target vector, $CR \in (0,1)$ = crossover rate, $r_j = rand() \in (0,1) \mid j = 1,2,3, \dots, D$

Using the mutated target vector 1 in Table 3, when *CR*=0.8, recombination sets a random number in (0,1) in every position, *e.g.*, 0.81, 0.88, 0.87, 0.92, 0.81, 0.88, 0.95, 0.99, 0.69, 0.46, 0.77. At position 9, r_j=0.69 was the first $r_j < CR$ in the trial vector. Thus, the mutant vector, $V_{ji,G+1}$, was copied into positions 1-9, and the target vector, $X_{ji,G+1}$, was copied into positions 10-11. Then, the trial vector was assigned to workstations without conflicts in the precedence diagram and the number of workstations, *m*. The results of the obtained trial vectors are shown in Table 3.

2.4 Selection

Selection chooses the target vector for the next round or selects from the population for the next generation, *G+1*, by comparing the solutions obtained from the target vector with those from the trial vector. If the solutions obtained from the trial vector are lower than or equal to those from the target vector, the trial vector will be selected for the next generation, else the target vector is selected as the next generation:

Table 3. Trial vector generated from target vector 1

	Workstation	1			2			3			4	
	Dimension	1	2	3	4	5	6	7	8	9	10	11
	Task	11	1	5	3	10	2	6	4	7	9	8
Vector 1	Target Vector	0.88	0.58	0.78	0.71	0.44	0.43	0.77	0.25	0.85	0.25	0.16
	Mutant Vector	0.78	0.74	0.67	0.61	0.54	0.51	0.54	0.37	0.34	0.22	0.10
	r_j	0.81	0.88	0.87	0.92	0.81	0.88	0.95	0.99	0.69	0.46	0.77
	Trial Vector	0.78	0.74	0.67	0.61	0.54	0.51	0.54	0.37	0.34	0.25	0.16

$$X_{ji,G+1} = \begin{cases} U_{ji,G+1} \; if \; f(U_{ji,G+1}) \le f(X_{ji,G+1}) \\ X_{ji,G} \; Other \end{cases} \quad (3)$$

where $X_{ji,G+1}$ = target vector and $U_{ji,G+1}$ = trial vector

On the basis of the explanations in sections 2.1-2.4, our modified DE is shown in Algorithm 1.

Our new DE algorithm was evaluated using data retrieved from Scholl [33], which included 15 problem sets (101 instances): 10 medium problem sets with 21-58 tasks (50 instances) and 5 large problem sets with 70-297 tasks (51 instances). The UALBP-2 was solved by applying the java program in NetBeans software environment on a computer with an Intel i5-8500 CPU@3.0 GHz and 4GB RAM. The parameters used here were used in the previous study [13], in which the DE algorithm was developed to solve U-shaped assembly line balancing problem - type 1 (UALBP-1). The parameters were *R*=30 rounds, *NP*=30, *F*=0.8, and *CR*=0.8.

Algorithm. Pseudo code of the DE for (UALBP-2)

Setup initial DE parameter: *NP*,*CR*,*F*,*D* (size of vector)
Begin randomly generate a set of target vector V_i *(i=1…NP);*
while termination condition is not satisfied **do**
for *i=1* to *NP* **do**
Perform mutation process using Eq. (1)
Perform recombination process using Eq. (2)
Perform selection process using Eq. (3)
End for
End while
Select best solution from all DE in the iteration
End DE
Show/select best solution from all DE in all iteration

3. Results and discussion

To assess the efficiency of our DE algorithm, our results were compared with those of the rule-based heuristic methods from Li et al. [5], who applied two-rule (task selection and task assignment) or three-rule (task selection, task assignment, and task exchange) approaches to solve UALBP-2. Medium problem results are shown in Table 4 and large ones in Table 5.

For both tables, column 2 shows the numbers of workstations for each problem; columns 3-6 show the optimal cycle time (OPT) and computational time (CPU in seconds) using the rules based heuristic methods [5]. Columns 7-8 show the same results for our new DE algorithm. Column 9 shows the cycle time difference (as %) between the DE algorithm and two rules method (PDCT-1) and column 10 shows the differences with three rules method (PDCT-2). Negative values indicate that our DE algorithm generated better cycle times, whereas positive values represent worse times. A "0" in the PDCT-1 or PDCT-2 columns indicates no difference.

Table 4 shows the minimized cycle times for 50 medium problems with 21-58 tasks. The results show that our DE algorithm generated a better cycle time than two rules and three rules for 14 instances (28%), the same cycle time as three rules but better than two rules for 19 instances (38%), a worse cycle time than three rules but better than two rules for 2 instances (4%), and the same cycle time as two rules and three rules for 14 instances (28%). It is obvious that our modified DE algorithm solves all problems better than using two rules. For the Hahm-53-9 problem, our method took 24% less time than the two rules. With three rules, our DE algorithm found solutions with equal and better results in 96% of the cases and only in two cases (4%) no better times were found. Moreover, our DE algorithm required less computation time than both the two and three rules for medium problems.

As for large problems, our DE algorithm was tested on 51 large problems containing 70-297 tasks. The minimum cycle times were compared with two rules and three rules, data from Li et al. [5], see Table

Table 4. Results for medium problems

Example	m	Rules-based heuristic [5]				Differential Evolution (DE)			
		Two rules		Three rules					
		OPT	CPU	OPT	CPU	OPT[1]	CPU	PDCT-1	PDCT-2
Mitchell-21	4	27	0.24	27	0.26	27	0.02	0	0
	5	22	0.3	21	0.1	21	0.03	-4.76	0
	6	18	0.31	18	0.37	18	0.03	0	0
Rosenberg-25	4	33	0.22	33	0.27	32	0.06	-3.13	-3.13
	5	25	0.14	25	0.11	25	0.03	0	0
	6	23	0.19	21	0.43	21	0.03	-9.52	0
	7	18	0.35	18	0.43	18	0.03	0	0
	8	16	0.43	16	0.5	16	0.04	0	0
Heskiaoff-28	6	171	1.16	171	1.27	171	0.03	0	0
	7	152	0.29	147	1.18	147	0.03	-3.40	0
	8	129	0.82	129	0.82	129	0.04	0	0
	9	117	0.63	116	0.58	116	0.05	-0.86	0
	10	109	0.27	108	0.77	108	0.06	-0.93	0
Buxey-29	5	69	0.23	66	0.5	65	0.02	-6.15	-1.54
	6	56	0.34	55	0.6	54	0.03	-3.70	-1.85
	7	50	0.24	48	0.55	47	0.03	-6.38	-2.13
	8	44	0.25	41	0.59	41	0.03	-7.32	0
	9	39	0.16	38	0.45	38	0.05	-2.63	0
	10	35	0.29	34	0.57	33	0.03	-6.06	-3.03
Sawyer-30	5	65	0.79	65	0.7	65	0.03	0	0
	6	55	0.8	54	0.17	54	0.03	-1.85	0
	7	48	0.82	47	0.73	47	0.04	-2.13	0
	8	41	0.71	41	0.84	41	0.05	0	0
Lutz1-32	6	2448	0.22	2404	0.49	2424*	0.03	-0.99	0.83
	7	2092	0.3	2092	0.37	2092	0.04	0	0
	8	1830	0.18	1816	0.38	1816	0.05	-0.77	0
	9	1644	0.27	1622	0.16	1624*	0.05	-1.23	0.12
	10	1474	0.3	1474	0.31	1474	0.06	0	0
Gunther-35	5	98	0.97	97	1.09	97	0.05	-1.03	0
	6	83	0.56	82	0.73	81	0.05	-2.47	-1.23
	7	71	0.65	70	0.75	69	0.05	-2.90	-1.45
	8	62	0.7	61	0.82	61	0.05	-1.64	0
Kilbridge & Wester-45	6	95	0.61	92	0.41	92	0.08	-3.26	0
	7	83	0.88	79	1.9	79	0.1	-5.06	0
	8	73	0.89	69	0.39	69	0.11	-5.80	0
	9	66	0.77	62	1.61	62	0.12	-6.45	0
	10	59	0.89	56	1.77	56	0.14	-5.36	0
Hahm-53	5	2948	0.23	2823	1.57	2820	0.09	-4.54	-0.11
	6	2420	0.36	2416	1.21	2416	0.12	-0.17	0
	7	2082	0.51	2013	1.71	2010	0.14	-3.58	-0.15
	8	1840	0.44	1827	1.21	1775	0.16	-3.66	-2.93
	9	2193	0.11	1775	0.15	1775	0.18	-23.55	0
Warnecke-58	10	162	0.92	155	2.66	155	0.24	-4.52	0
	11	149	0.96	143	2.59	143	0.26	-4.20	0
	12	136	0.76	131	2.29	131	0.28	-3.82	0
	13	127	0.95	122	2.15	120	0.31	-5.83	-1.67
	14	115	1.27	112	2.14	111	0.33	-3.60	-0.90
	15	110	0.88	105	2.5	104	0.35	-5.77	-0.96
	16	103	1.03	98	2.1	97	0.37	-6.19	-1.03
	17	96	1.34	94	1.83	92	0.4	-2.13	0

Remarks[1]: **0** = better cycle time than two rules and three rules

0 = the same cycle time as three rules, but better cycle time than two rules

0* = worse cycle time than three rules, but better cycle time than two rules

0** = worse cycle time than three rules, but the same cycle time as two rules

Table 5. Results for large problems

Example	*m*	Rules-based heuristic [5]				Differential Evolution (DE)			
		Two rules		Three rules					
		OPT	CPU	OPT	CPU	OPT[1]	CPU	PDCT-1	PDCT-2
Tonge-70	10	359	1.54	352	3.87	359**	0.39	0	1.95
	11	323	2.03	320	3.28	320	0.44	-0.94	0
	12	298	2.19	295	2.9	293	0.46	-1.71	-0.68
	13	275	1.14	273	2.15	275**	0.54	0	0.73
	14	255	1.45	252	2.46	255**	0.55	0	1.18
	15	237	1.26	236	2	237**	0.6	0	0.42
	16	232	1.24	222	1.52	232**	0.61	0	4.31
Lutz2-89	13	40	1.91	38	2.88	38	0.95	-5.26	0
	14	37	1.92	35	2.89	35	1.03	-5.71	0
	15	33	2.74	33	2.78	33	1	0	0
	16	31	2.57	31	2.62	31	1.17	0	0
	17	30	2.38	30	2.45	29	1.24	-3.45	-3.45
	18	29	1.72	27	2.73	27	1.32	-7.41	0
	19	27	2.42	27	2.32	26	1.4	-3.85	-3.85
	20	27	1.71	25	2.65	25	1.47	-8.00	0
	21	24	2.59	24	2.76	24	1.56	0	0
Mukherjee-94	14	307	3.67	302	8.12	307**	0.66	0	1.63
	15	284	6.12	283	6.92	283	0.69	-0.35	0
	16	267	3.65	265	6.31	265	0.72	-0.75	0
	17	252	3.01	251	5.71	251	0.79	-0.40	0
	18	237	4.55	236	6.05	236	0.85	-0.42	0
	19	224	4.03	224	4.44	224	0.9	0	0
	20	223	2.85	212	5.37	221*	0.95	-0.90	4.07
	21	211	4.47	205	8.07	210*	0.99	-0.48	2.38
	22	203	2.04	193	4.16	201*	1.03	-1.00	3.98
	23	193	1.64	185	7.54	193**	1.09	0	4.15
	24	186	2.47	179	7.24	186**	1.14	0	3.76
Bartholdi-148	7	806	19.04	805	20.77	805	2.19	-0.12	0
	8	712	12	705	20.23	705	2.51	-0.99	0
	9	628	16.38	626	11.33	626	2.85	-0.32	0
	10	564	14.3	564	15.87	564	3.14	0	0
	11	520	10.27	513	13.68	513	3.46	-1.36	0
	12	475	10.03	470	10.51	470	3.79	-1.06	0
	13	437	11.12	434	24.6	434	4.1	-0.69	0
	14	405	12.58	403	22.35	403	4.46	-0.50	0
Scholl-297	21	3490	10.13	3328	35.65	3460*	20.1	-0.87	3.82
	23	3189	10.27	3034	53.13	3133*	21.65	-1.79	3.16
	25	2931	11.32	2790	47.36	2925*	22.57	-0.21	4.62
	27	2695	25.45	2582	69.36	2684*	24.26	-0.41	3.80
	29	2526	14.11	2406	48.9	2519*	25.46	-0.28	4.49
	31	2366	10.67	2256	43.26	2358*	25.25	-0.34	4.33
	33	2203	16.08	2114	56.53	2144*	27.55	-2.75	1.40
	35	2011	41.62	1993	62.17	2001*	28.1	-0.50	0.40
	37	1983	13.43	1888	45.6	1980*	29.22	-0.15	4.65
	39	1882	14.91	1792	40.66	1880*	31.02	-0.11	4.68
	41	1779	17.18	1702	58.57	1774*	32.22	-0.28	4.06
	43	1706	13.87	1623	36.72	1697*	33.12	-0.53	4.36
	45	1620	12.98	1551	53.3	1616*	37.55	-0.25	4.02
	47	1562	8.67	1488	42.92	1544*	38.45	-1.17	3.63
	49	1496	15.44	1438	34.72	1479*	39.25	-1.15	2.77
	50	1458	12.13	1398	40.51	1421*	40.13	-2.60	1.62

Remarks[1]: **0** = better cycle time than two rules and three rules

0 = the same cycle time as three rules, but better cycle time than two rules

0* = worse cycle time than three rules, but better cycle time than two rules

0** = worse cycle time than three rules, but the same cycle time as two rules

5. The DE algorithm generated better cycle time than both the two and three rules methods for 3 instances (6%), the same cycle time as three rules but better than two rules for 16 instances (31%), worse cycle time than three rules but better than two rules for 19 instances (37%), worse cycle time than three rules but the same as two rules for 8 instances (16%), and the same cycle time as both two rules and three rules methods for 5 instances (10%).

Comparing the computational time with two rules, our DE algorithm was shorter for 37 instances (73%) and was longer for 14 instances (27%), and these 14 instances were in the Scholl-297 problems with high number of workstations, ranging from 21 to 50. With three rules, the DE algorithm was faster for 50 instances (98%) and slower for only a single instance (2%).

From Tables 4 and 5, we can observe that the DE algorithm provides comparable or better solutions than the two rules methods for all instances of medium and large problems. However, for the three rules, there were 29 from 101 instances, in which the DE algorithm did not find better or comparable cycle times. When examining these 29 instances, 27 instances, ~93%, were large problems with a high number of tasks and complicated precedence. Thus, further study to explore this issue, particularly in large problems, is needed.

4. Conclusions

Minimum cycle times in U-shaped assembly line balancing problem - type 2 (UALBP-2) were measured using our modified DE algorithm for 50 medium problems, with 21-58 tasks and 51 large problems, with 70-297 tasks. All 101 instances could be solved and found the best solutions. The minimum cycle times obtained from this approach were also compared with those from the rule-based heuristic methods, with two rules or three rules, used by Li et al. [5]. Our DE algorithm found the best solutions for both medium and large problems faster. When analyzing the optimal solutions, the DE algorithm was able to find the best, better, or equivalent solutions in almost all instances for medium-scale problems (48 from 50 instances or 96%). However, the proposed approach failed to find the best solutions in the large-scale problems for approximately 54% (27 from 51 instances). This is due to the number of tasks and precedence relationship conditions increased and more complicated. Thus, our DE algorithm is a potential and effective approach to solve the UALBP-2, particularly in medium-scale problems.

In future work, we will improve our DE algorithm for better solutions in large problems and verify our expectation that the DE algorithm can efficiently solve other assembly line balancing problems, for example, Mixed/Multi line, Parallel line, Two-Sided Line and others.

Conflicts of interest

The authors confirm that this publication has no established conflicts of interest and that there has been no substantial financial funding for this work that may have influenced its result.

Author contributions

The contributions of authors are as follows: conceptualization, K. Chantarasamai; methodology, K. Chantarasamai; software, K. Chantarasamai; validation, K. Chantarasamai and O. Lasunon; formal analysis, K. Chantarasamai; investigation, K. Chantarasamai; resources, K. Chantarasamai; data curation, K. Chantarasamai; writing—original draft preparation, K. Chantarasamai; writing—review and editing, K. Chantarasamai and O. Lasunon; visualization, O. Lasunon; supervision, O. Lasunon; project administration, O. Lasunon. All authors read and approved the final manuscript.

Acknowledgments

We would like to thank the Manufacturing and Material Research Unit, Faculty of Engineering, Mahasarakham University for providing research equipment.

References

[1] N. Kriengkorakot and N. Pianthong, “The Assembly Line Balancing Problem: Review Articles”, *KKU Engineering Journal*, Vol. 34, No. 2, pp. 133-140, 2007.

[2] G. Jirasirilerd, R. Pitakaso, K. Sethanan, S. Kaewman, W. Sirirak, and M. Kosacka-Olejnik, “Simple Assembly Line Balancing Problem Type 2 by Variable Neighborhood Strategy Adaptive Search: A Case Study Garment Industry”, *Journal of Open Innovation: Technology, Market, and Complexity*, Vol. 6, No. 1, pp. 1-22, 2020.

[3] N. Kriengkorakot and N. Pianthong, “The U-Line Assembly Line Balancing Problem”, *KKU Engineering Journal*, Vol. 34, No. 3, pp. 267-274, 2007.

[4] V. Jonnalagedda and B. Dabade, “Application of Simple Genetic Algorithm to U-Shaped

Assembly Line Balancing Problem of Type II", In: *Proc. of International Conf. On Automatic Control*, Cape Town, South Africa, pp. 6168-6173, 2014.

[5] M. Li, Q. Tang, Q. Zheng, X. Xia, and C. A. Floudas, "Rules-Based Heuristic Approach for the U-Shaped Assembly Line Balancing Problem", *Applied Mathematical Modelling*, Vol. 48, pp. 423-439, 2017.

[6] G. J. Miltenburg and J. Wijngaard, "The U-line Line Balancing Problem", *Management Science*, Vol. 40, No. 10, pp. 1378-1388, 1994.

[7] T. L. Urban, "Note. Optimal Balancing of U-Shaped Assembly Lines", *Management Science*, Vol. 44, No. 5, pp. 738-741, 1998.

[8] A. Scholl and R. Klein, "ULINO: Optimally Balancing U-shaped JIT Assembly Lines", *International Journal of Production Research*, Vol. 37, No. 4, pp. 721-736, 1999.

[9] R. K. Hwang, H. Katayama, and M. Gen, "U-Shaped Assembly Line Balancing Problem with Genetic Algorithm", *International Journal of Production Research*, Vol. 46, No. 16, pp. 4637-4649, 2008.

[10] A. Baykasoğlu and T. Dereli, "Simple and U-type Assembly Line Balancing by using an Ant Colony Based Algorithm", *Mathematical and Computational Applications*, Vol. 14, No. 1, pp. 1-12, 2009.

[11] M. Fathi, M. J. Alvarez, and V. Rodríguez, "The U-line Line Balancing Problem", *World Academy of Sciences Journal*, Vol. 5, No. 11, pp. 2115-2123, 2011.

[12] N. Kriengkorakot, and P. Kriengkorakot, "Heuristics Comparison for U-Shaped Assembly Lin Balancing in the Apparel Factory", *KKU Engineering Journal*, Vol.41, No.2, pp.155-162, 2014.

[13] P. Sresracoo, N. Kriengkorakot, P. Kriengkorakot, and K. Chantarasamai, "U-Shaped Assembly Line Balancing by Using Differential Evolution Algorithm", *Mathematical and Computational Applications*, Vol. 23, No. 4, pp. 1-21, 2018.

[14] 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 Computing and Applications*, Vol. 31, pp. 7501-7515, 2019.

[15] Y. Li, X. Hu, X. Tang, and I. Kucukkoc, "Type-1 U-shaped Assembly Line Balancing under uncertain task time", In: *Proc. of International Conf. On Manufacturing Modelling*, Berlin, Germany, pp. 992-997, 2019.

[16] S. Varnasilpin and R. Masuchun, "The Allowable Time Approach of the Uncertain Task for Three U-shaped Lines with the Minimum Workstations", *International Journal of Intelligent Engineering and Systems*, Vol. 13, No. 1, pp.203-213, 2020.

[17] R. Klein and A. Scholl, "Maximizing the Production Rate in Simple Assembly Line Balancing - A Branch and Bound Procedure", *European Journal of Operational Research*, Vol. 91, No. 2, pp. 367-385, 1996.

[18] S. B. Liu, H. L. Ong, and H.C. Huang, "Two Bi-Directional Heuristics for the Assembly Line Type II Problem", *The International Journal of Advanced Manufacturing Technology*, Vol. 22, pp. 656-661, 2003.

[19] S. B. Liu, H. L. Ong, and H. C. A. Huang, "Bidirectional Heuristic for Stochastic Assembly Line Balancing Type II Problem", *The International Journal of Advanced Manufacturing Technology*, Vol. 25, pp. 71-77, 2005.

[20] S. A. Seyed-Alagheband, S. M. T. Fatemi Ghomi, and M. A. Zandieh, "Simulated Annealing Algorithm for Balancing the Assembly Line Type II Problem with Sequence-Dependent Setup Times between Tasks", *International Journal of Production Research*, Vol. 49, No. 3, pp. 805-825, 2011.

[21] D. Karaboğa, and S. Ökdem, "A Simple and Global Optimization Algorithm for Engineering Problems: Differential Evolution Algorithm", *Turkish Journal of Electrical Engineering and Computer Sciences*, Vol. 12, No. 1, pp. 53-60, 2004.

[22] S. Kaewman, T. Srivarapongse, C. Theeraviriya, and G. Jirasirilerd, "Differential Evolution Algorithm for Multilevel Assignment Problem: A Case Study in Chicken Transportation", *Mathematical and Computational Applications*, Vol. 23, No. 4, pp. 1-19, 2018.

[23] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces", *Journal of Global Optimization*, Vol. 11, pp. 341-359, 1997.

[24] U. Ketsripongsa, R. Pitakaso, K. Sethanan, and T. Srivarapongse, "An Improved Differential Evolution Algorithm for Crop Planning in the Northeastern Region of Thailand", *Mathematical and Computational Applications*, Vol. 23, No. 3, pp. 1-19, 2018.

[25] R. Kamphukaew, K. Sethanan, T. Jamrus, and H. K. Wang, "Differential Evolution Algorithms with Local Search for the Multi-Products

Capacitated Vehicle Routing Problem with Time Windows: A Case Study of the Ice Industry", *Engineering and Applied Science Research*, Vol. 45, No. 4, pp. 273-281, 2018.

[26] P. Chokanat, R. Pitakaso, and K. Sethanan, "Methodology to Solve a Special Case of the Vehicle Routing Problem: A Case Study in the Raw Milk Transportation System", *AgriEngineering*. Vol. 1, No. 1, pp. 75-93, 2019.

[27] R. Akararungruangkul and S. Kaewman, "Modified Differential Evolution Algorithm Solving the Special Case of Location Routing Problem", *Mathematical and Computational Applications*, Vol. 23, No. 3, pp. 1-16, 2018.

[28] R. Pitakaso, "Differential Evolution Algorithm for Simple Assembly Line Balancing Type 1 (SALBP-1)", *Journal of Industrial and Production Engineering*, Vol. 32, No. 2, pp. 104-114, 2015.

[29] A. C. Nearchou and S. L. Omirou, "Assembly Line Balancing Using Differential Evolution Models", *Cybernetics and Systems*, Vol. 48, No. 5, pp. 436-458, 2017.

[30] M. Ramadas, A. Abraham, and S. Kumar, "ReDE- A Revised mutation strategy for Differential Evolution Algorithm", *International Journal of Intelligent Engineering and Systems*, Vol. 9, No. 4, pp. 51-58, 2016.

[31] T. Srichok, R. Pitakaso, K. Sethanan, W. Sirirak, and P. Kwangmuang, "Combined Response Surface Method and Modified Differential Evolution for Parameter Optimization of Friction Stir Welding", *Processes*, Vol. 8, No. 9, pp. 1-22, 2020.

[32] J. R. Jackson, "A Computing Procedure for a Line Balancing Problem", *Management Science*, Vol. 2, No. 3, pp. 261-271, 1956.

[33] A. Scholl, "Data of Assembly Line Balancing Problems", 1993, [Online]. Available: https:assembly-line-balancing.de/salbp/benchmark-data-sets-1993.