A Hybrid Differential Evolution Algorithm with Local Search for Optimizing Cycle Time in U-Shaped Assembly Line Balancing Problem Type 2

Krit Chantarasamai
Department of Mechanical Engineering, Faculty of Agriculture and Technology, Rajamangala University of Technology Isan, Surin Campus, Thailand
krit.ch@rmuti.ac.th

Visit Junchuan
Department of Mechanical Engineering, Faculty of Agriculture and Technology, Rajamangala University of Technology Isan, Surin Campus, Thailand
visit.ju@rmuti.ac.th

Poontana Sresracoo
Department of Industrial Management Engineering, Faculty of Industrial Technology, Buriram Rajabhat University, Thailand
phoonthana.ss@bru.ac.th (corresponding author)

Received: 2 July 2025 | Revised: 13 August 2025, 22 August 2025, 1 September 2025, 11 September 2025, 30 September 2025, and 5 October 2025 | Accepted: 9 October 2025

 DOI: https://doi.org/10.48084/etasr.13120

ABSTRACT

U-Shaped Assembly Line Balancing Problem Type 2 (UALBP-2) is a key challenge in modern Just-In-Time (JIT) manufacturing, where the objective is to produce goods quickly while keeping waiting times as short as possible. This study introduces a new problem-solving method called the Hybrid Differential Evolution Algorithm with Local Search (HDE), which combines an existing optimization approach, Differential Evolution (DE), with an additional local search technique to improve results. The method was tested against three common approaches: a mathematical model, a Genetic Algorithm (GA), and rule-based heuristic methods, using well-known test problems from the literature. The results showed that the HDE algorithm often outperformed these other methods: it gave better solutions than the mathematical model in 40% of all instances, better than the GA in 84% of all instances, and better than rule-based heuristic methods in about 40% of medium- and large-scale problems. Overall, the findings indicate that the HDE algorithm is a very effective tool for minimizing production time in U-shaped assembly lines, making it a promising option for industries that rely on JIT manufacturing.

Keywords-U-Shaped Assembly Line Balancing Problem (UALBP); Just-In-Time (JIT) manufacturing; Hybrid Differential Evolution Algorithm with Local Search (HDE); optimization algorithms; production efficiency

I. INTRODUCTION

With the advent of Industry 4.0, companies have adopted digital technologies to improve efficiency and reduce costs. These tools also help meet customer demands under the Just-In-Time (JIT) principle. In this context, Assembly Line Balancing (ALB) plays a central role. Several studies illustrate this trend, such as the use of robots in U-shaped lines [1, 2], the application of heuristic and artificial neural network methods in the lighting industry [3], and the integration of the Internet of Things (IoT) for line balancing [4]. These methods confirm the importance of ALB for JIT systems; however, the literature highlights that the existing algorithms are not yet highly efficient. Therefore, the development of algorithms integrated with digital technologies can enhance efficiency and is a key driver of productivity in Industry 4.0.

The U-shaped Assembly Line Balancing Problem (UALBP), introduced in 1994 [5], is a variant of the General Assembly Line Balancing Problem (GALBP) [6, 7]. It is particularly attractive in modern manufacturing, as compared with straight lines, it requires less space, fewer workstations,

and fewer operators, while offering greater flexibility [8]. The UALBP is typically classified into three types: UALBP-1, which minimizes the number of workstations; UALBP-2, which minimizes cycle time; and UALBP-E, which maximizes line efficiency. Since its introduction, a wide range of approaches have been proposed, including dynamic programming, bi-objective simulated annealing [9], enhanced beam search heuristic [10], Genetic Algorithm (GA) [11, 12], Differential Evolution (DE) algorithm [13, 14], rule-based heuristics [15], and branch-and-bound methods [16]. These approaches have also found real-world applications, such as in apparel manufacturing [17] and in water meter production under task time uncertainty [18], demonstrating the practical relevance of the UALBP.

Among these variants, UALBP-2 is particularly critical, since minimizing cycle time directly determines the ability to deliver products on schedule, which lies at the core of JIT. Various algorithms have been developed to address it. GA [12] performs well but is limited to small-scale problems. Rule-based heuristics [15] are more effective on medium- and large-scale problems, but they do not guarantee optimal solutions. DE [14] is appreciated for its computational efficiency and promising results on medium-sized problems, but its performance deteriorates significantly when tackling large-scale problems.

Since its introduction in 1997 [19], DE has proven to be a versatile optimization tool across diverse domains, ranging from job-shop scheduling [20-22] to assignment and transportation problems [23-25], location-routing [26], and even image segmentation [27]. Its effectiveness in ALB has also been demonstrated, for Simple Assembly Line Balancing Problem Type 1 (SALBP-1) [28-31], UALBP-1 [13, 32], and UALBP-2 [14]. However, findings in the literature highlight that DE struggles to maintain solution quality as problem size increases.

This limitation highlights the need for hybrid approaches. In particular, integrating a local search technique allows solutions generated by DE to be progressively refined, thereby improving its ability to explore the solution space effectively. Motivated by this, the present study aims to develop a Hybrid Differential Evolution Algorithm with Local Search (HDE) to achieve higher-quality solutions for large-scale UALBP-2 instances. The performance of the proposed algorithm will be evaluated by comparing it with a mathematical model, a GA, a rule-based heuristic, and an existing Modified Differential Evolution (MDE) algorithm.

II. MATHEMATICAL MODEL

This section presents the mathematical model of the UALBP-2 problem. The objective function (1) aims to minimize the cycle time. Condition (2) ensures that each task is assigned to exactly one workstation. The cycle time condition (3) guarantees that the total processing time at any workstation does not exceed the cycle time. Conditions (4) and (5) represent the precedence relationships, ensuring that no task is assigned to an earlier workstation than its predecessor.

The indices, parameters, and decision variables are defined as follows:

Indices:

- i denotes the index of task i, where $i = 1, \dots, n$
- j denotes the index of task j, while $j = 1, \dots, M_{max}$
- n denotes the total number of tasks
- r, s denote tasks in a precedence relationship, with r preceding s
- M_{max} denotes the total number of workstations

Parameters:

- C denotes the cycle time of a workstation
- t_i denotes the processing time of task i
- P denotes the set of ordered pairs of tasks reflecting the precedence relationships

Decision variables:

- $X_{ij} = 1$ if task i of the original network is assigned to workstation j; 0 otherwise
- $Y_{ij} = 1$ if task i of the phantom network is assigned to workstation j; 0 otherwise

The objective function is:

$$\text{Minimize } C \quad (1)$$

subject to:

$$\sum_{j=1}^{M_{max}} (X_{ij} + Y_{ij}) = 1, \quad \forall i = 1, \dots, n \quad (2)$$

$$\sum_{i=1}^{n} t_i (X_{ij} + Y_{ij}) \leq C, \quad \forall j = 1, \dots, M_{max} \quad (3)$$

$$\sum_{j=1}^{M_{max}} (M_{max} - j + 1)(X_{rj} - X_{sj}) \geq 0, \ \forall (r,s) \in P \quad (4)$$

$$\sum_{j=1}^{M_{max}} (M_{max} - j + 1)(Y_{rj} - Y_{sj}) \geq 0, \ \forall (r,s) \in P \quad (5)$$

where $X_{ij}, Y_{ij} \in \{0,1\}, \ \forall i, j$.

III. PROPOSED HEURISTICS

The HDE operates in five main steps: initial solution generation, mutation, recombination, local search, and selection. These steps are illustrated in Figure 1.

A. Initial Solution Generation

In this step, an initial solution for each task is generated randomly using numbers between 0 and 1. These random numbers are then used to assign tasks to workstations. The initial solution generated is called the "target vector." The number of workstations, m, and the population size (number of population), NP, are initialized before the algorithm starts. Each initial vector V_i, where $i = 1, \dots, NP$, consists of D random numbers: $V_i = \{r_1, \dots, r_D\}$, where $r_j = [1,0]$, $j = 1, \dots, D$.

B. Mutation

In the mutation step, values within each dimension (D) of the vectors are adjusted to obtain new solutions, referred to as "mutant vectors." The mutant vector $V_{i,G+1}$ is calculated using

(6), where $i \in \{1, NP\}$ is the vector index and $G \in \{1, R\}$ is a generation number:

$$V_{i,G+1} = X_{best,G} + F\left(X_{r1,G} - X_{r2,G}\right) + F\left(X_{r3,G} - X_{r4,G}\right) \quad (6)$$

where $X_{best,G}$ is the best vector, $X_{r1,G}$, $X_{r2,G}$, $X_{r3,G}$, $X_{r4,G}$ are random vectors from the initial solutions NP, and F is the scaling factor.

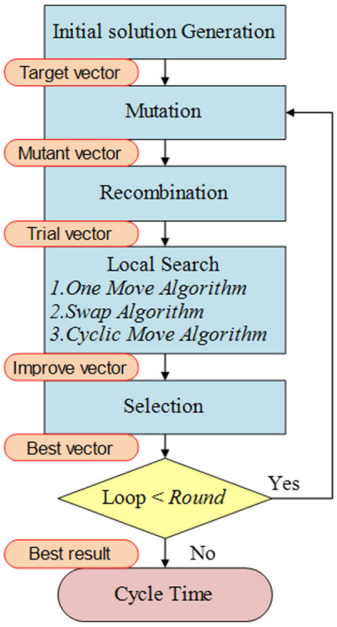

Fig. 1. Flowchart of the HDE algorithm for UALBP-2.

C. Recombination

During the recombination step, the target and mutant vectors are combined using the crossover rate CR. The resulting vector is called the "trial vector," which is evaluated using the exponential crossover (one-position method) as defined in (7):

$$U_{ji,G+1} = \begin{cases} V_{ji,G+1}, & if\ r_j < CR \\ X_{ji,G}, & otherwise \end{cases} \quad (7)$$

where $U_{ji,G+1}$ is the trial vector, $V_{ji,G+1}$ is the mutant vector, $X_{ji,G}$ is the target vector, $CR \in (0,1)$ is the crossover rate, and $r_j = rand() \in (0,1)$ for $j = 1,2,\ldots,D$.

D. Local Search

This section outlines the improvement of the DE algorithm for UALBP-2 by adding a local search algorithm after the recombination process. The local search methods considered include the one-move, swap, and cyclic-move algorithms.

1) One-Move Algorithm

This method is used to improve a heuristic-based solution by selecting one task from a workstation and moving it to another (the next) workstation. For example, if Task 3 is randomly selected from Workstation 2, its move to Workstation 1 must not violate task precedence constraints or exceed the cycle time. This task reassignment aims to reduce the cycle time, as illustrated in Figure 2.

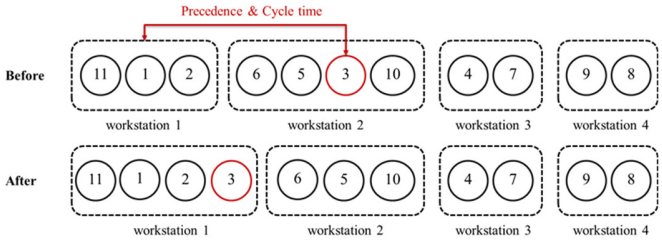

Fig. 2. Example of improvement using the one-move algorithm.

2) Swap Algorithm

This method aims to improve the solution by swapping the positions of a pair of tasks between two workstations. For instance, if Task 3 is initially assigned to Workstation 2, the algorithm swaps it with Task 7 from Workstation 3. This task swap must not violate any precedence constraints or exceed the predefined cycle time. Such swaps are intended to reduce the overall cycle time, as illustrated in Figure 3.

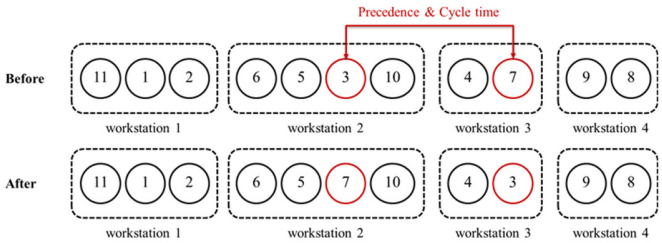

Fig. 3. Example of improvement using the swap algorithm.

3) Cyclic-Move Algorithm

This method selects one task from each workstation and then cyclically shifts these tasks among the workstations. For example, Task 8, originally assigned to Workstation 4, is moved to Workstation 3. Subsequently, Task 7 from Workstation 3 is moved to Workstation 2, and Task 3 from Workstation 2 is moved to Workstation 1. Finally, Task 11 from Workstation 1 is moved to Workstation 4, completing one cycle and relocating all involved tasks. This process can be repeated for multiple rounds, with tasks randomized in each round. These cyclic shifts aim to reduce the cycle time, as illustrated in Figure 4.

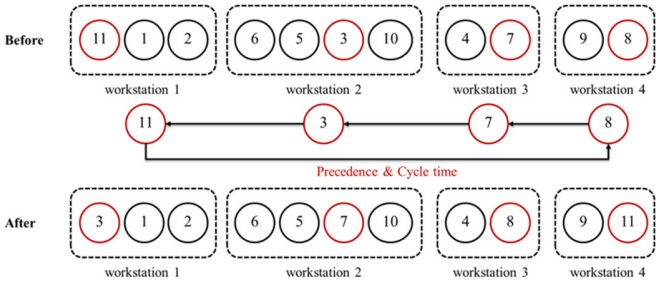

Fig. 4. Example of improvement using the cyclic-move algorithm.

E. Selection

In the selection step, the target vector for the next generation $(G+1)$ is determined by comparing its objective function value with that obtained from the trial vector. The vector yielding the superior result is then selected for the next evaluation. If the trial vector's result is not superior to that of the target vector, the target vector is retained for the subsequent generation.

$$X_{ji,G+1} = \begin{cases} U_{ji,G+1}, & if\ f(U_{ji,G+1}) \le f(X_{ji,G+1}) \\ X_{ji,G}, & otherwise \end{cases} \quad (8)$$

where $X_{ji,G+1}$ is the target vector and $U_{ji,G+1}$ is the trial vector.

Based on the explanations in Section III, the pseudocode of the HDE algorithm is shown in Figure 5.

```
Procedure: Hybrid Differential Evolution with local search (HDE)
Input: task, workstation, precedence relationship, cycle time,
       differential evolution parameter (NP, CR, F, m)
Output: Optimal solution
Begin: randomly generate a set of target vector Vi (i=1...NP);
    while termination condition is not satisfied do
        for i=1 to NP do
            Mutation Process
            Recombination Process
            Local Search Process
            Selection Process
        end
    end // Select the best solution from all HDE in the iteration
end; // Select the best solution from all HDE in all iterations
```

Fig. 5. Pseudocode of the HDE algorithm.

IV. RESULTS AND DISCUSSION

This section explores the benefits of the proposed HDE by comparing its efficiency with three existing methods from the literature. First, the results of the HDE algorithms are compared with those obtained from a mathematical model solved using Lingo V.11. Second, its performance is benchmarked against a GA [12]. Finally, a comparison is made with rule-based heuristic methods [15].

The HDE algorithm was evaluated using benchmark datasets provided by Scholl [33], which include problems with 8 to 297 tasks. The UALBP-2 was solved by implementing a Java program in NetBeans software on a computer equipped with an Intel i7-3630QM CPU @ 2.40 GHz and 8 GB RAM. The parameters used in this study were set as follows: $R = 30$ rounds, $NP = 30$, $F = 0.8$, and $CR = 0.8$.

In Tables I-IV, the test instances are presented with columns detailing: number of workstations (m), optimal cycle time (OPT), and computational time (CT, seconds). The heuristics methods column includes: Lingo, GA, rule-based heuristics (Two Rules and Three Rules), MDE, and HDE.

A. Comparison with the Mathematical Model

The computational results comparing the developed heuristics with the mathematical model solved using Lingo V.11 are presented in Table I. This comparison uses Scholl's benchmark dataset, which includes 38 instances with the number of tasks ranging from 8 to 45. The experiment concludes upon the completion of the specified number of rounds.

TABLE I. COMPARISON OF MATHEMATICAL MODEL, MDE, AND HDE RESULTS

Example [33]	m	Lingo		MDE [14]		HDE	
		OPT	CT	OPT	CT	OPT	CT
Bowman-8	2	38	0.01	38	0.02	38	0.01
	3	26	0.01	26	0.01	26	0.01
	4	20	0.01	20	0.01	20	0.01
Jackson-11	2	23	0.01	23	0.03	23	0.01
	3	16	0.01	16	0.02	16	0.02
	4	12	0.01	12	0.03	12	0.03
Mitchell-21	4	27	0.01	27	0.02	27	0.02
	5	21	0.01	21	0.03	21	0.02
	6	18	0.01	18	0.03	18	0.03
Rosenberg-25	4	33	0.01	32	0.06	32	0.03
	5	26	0.06	25	0.03	25	0.02
	6	22	0.02	21	0.03	21	0.03
	7	19	0.02	18	0.03	18	0.03
	8	17	0.08	16	0.04	16	0.03
Heskiaoff-28	6	172	0.02	171	0.03	171	0.03
	7	147	0.10	147	0.03	147	0.03
	8	129	0.04	129	0.04	128	0.03
	9	115	0.42	116*	0.05	114	0.03
	10	108	0.06	108	0.06	108	0.06
Buxey-29	5	67	0.09	65	0.02	65	0.03
	6	56	0.07	54	0.03	54	0.03
	7	48	0.10	47	0.03	47	0.03
	8	42	0.25	41	0.03	41	0.03
	9	38	0.24	38	0.05	36	0.03
	10	34	2.01	33	0.03	33	0.05
Sawyer-30	5	65	0.01	65	0.03	65	0.03
	6	54	0.03	54	0.03	54	0.03
	7	47	0.02	47	0.04	47	0.05
	8	41	0.15	41	0.05	41	0.04
Gunther-35	5	97	0.03	97	0.05	97	0.03
	6	81	0.14	81	0.05	81	0.05
	7	69	14.02	69	0.05	69	0.05
	8	61	1.23	61	0.05	61	0.05
Kilbridge&Wester-45	6	92	0.08	92	0.08	92	0.07
	7	79	0.02	79	0.10	79	0.09
	8	69	13.42	69	0.11	69	0.10
	9	62	0.05	62	0.12	62	0.11
	10	56	0.21	56	0.14	56	0.13
Average computational time			0.87		0.04		0.04

Remarks: * indicates a worse cycle time than Lingo, and values without an asterisk indicate that the cycle time is equal to or better than Lingo.

As shown in Table I, the HDE algorithm demonstrates superior solution quality compared to both Lingo V.11 and the MDE algorithm. Specifically, the HDE algorithm finds a better cycle time for 15 instances (40%) compared with Lingo V.11, and for 3 instances (8%) compared to the MDE algorithm. Furthermore, the average computational time for both the HDE and MDE algorithms is significantly faster than that of the mathematical model solved by Lingo V.11, averaging only 0.04 s.

B. Comparison with the Genetic Algorithm

In this section, the HDE algorithm is compared with the GA [12] and the MDE algorithm [14]. This comparison utilizes problem datasets from Scholl, which comprises 25 instances with the number of tasks ranging from 11 to 70. The experiment terminates upon the completion of the specified number of rounds. The computational results are presented in Table II.

TABLE II. COMPARISON OF GA, MDE, AND HDE RESULTS

Example [33]	m	GA [12]	MDE [14]		HDE	
		OPT	OPT	CT	OPT	CT
Jackson-11	4	12	12	0.03	12	0.03
	3	16	16	0.02	16	0.02
Mitchell-21	8	15	14	0.02	14	0.02
	7	16	15	0.03	15	0.02
	6	18	18	0.03	18	0.03
	5	21	21	0.03	21	0.03
Heskiaoff-28	8	144	129	0.04	128	0.03
	7	159	147	0.03	147	0.03
	6	177	171	0.03	171	0.03
	5	208	205	0.03	205	0.03
Sawyer-30	11	35	30	0.03	30	0.03
	9	42	36	0.03	36	0.03
	7	49	47	0.04	47	0.05
	5	68	65	0.03	65	0.03
Kilbridge& Wester-45	10	63	56	0.14	56	0.13
	8	75	69	0.11	69	0.10
	6	96	92	0.08	92	0.07
	4	140	138	0.07	138	0.07
	3	185	184	0.07	184	0.08
Tong-70	10	382	359	0.39	351	0.39
	9	419	390	0.34	390	0.35
	8	457	439	0.33	439	0.34
	7	521	502	0.32	502	0.33
	6	607	585	0.32	585	0.31
	5	724	702	0.27	702	0.28
Average computational time				0.11		0.11

Remarks: * indicates a worse cycle time than GA, and values without an asterisk indicate that the cycle time is equal to or better than GA.

Table II demonstrates that both the MDE and HDE algorithms outperform the GA algorithm. Specifically, the HDE and MDE algorithms achieve better cycle times than the GA algorithm in 21 instances (84%). For two specific instances, the HDE algorithm finds a superior optimal cycle time compared to the MDE algorithm. Across this group of test instances, the average computational time for both the HDE and MDE algorithms is identical, at 0.11 s.

C. Comparison with Rule-Based Heuristic

In this section, the HDE algorithm is compared with the rule-based heuristic [15] and the MDE algorithm [14]. The rule-based heuristic is categorized into two methods: Two Rules (task selection and task assignment) and Three Rules (task selection, task assignment, and task exchange). The dataset contains 115 instances, with task numbers ranging from 21 to 297, divided into two categories: medium-scale problems (21–70 tasks, 57 instances) and large-scale problems (89–297 tasks, 58 instances). The experiment terminates upon the completion of the specified number of rounds. The computational results for medium-scale problems are presented in Table III, and those for large-scale problems in Table IV.

For medium-scale problems (21–70 tasks), the HDE algorithm outperforms both the rule-based heuristic and the MDE algorithm (Table III). Specifically, the HDE algorithm achieves better cycle times than the rule-based heuristic, for both the two-rule and three-rule variants, in 22 instances (39%), compared with 16 instances (28%) for the MDE algorithm. For the two-rule method, both HDE and MDE consistently find solutions that are equal to or better than the existing optimal solutions in 100% of instances. For the three-rule method, the MDE algorithm achieves equal or better results in 52 instances (88%), failing in only 7 instances (12%). The HDE algorithm achieves equal or better results in all instances (100%).

For large-scale problems (89–297 tasks), the HDE algorithm is compared with both the rule-based heuristic and the MDE algorithm (Table IV). The results indicate that the HDE algorithm achieves better optimal cycle times than the rule-based heuristic, for both the two-rule and three-rule variants, in 23 instances (40%), whereas MDE does so in only 2 instances (3%). For both two-rule and three-rule methods, the HDE algorithm consistently finds solutions equal to or better than existing optimal solutions in all instances (100%). In contrast, the MDE algorithm finds solutions equal to or better than two rules in all instances but only in 25 instances (43%) (black) for three rules.

Regarding computational time (CT), for medium-scale problems (Table III), both the HDE and MDE algorithms averaged 0.15 s per instance. In terms of relative performance, HDE and MDE were faster in 50 instances (98%) and slower in only one instance (2%). For large-scale problems (Table IV), the HDE algorithm required an average of 12.32 s per instance. This is shorter than the computational time of the rule-based heuristics, indicating higher efficiency, but slightly higher than that of the MDE algorithm. Specifically, when using two rules, both HDE and MDE were faster in 43 instances (74%) and slower in 15 instances (26%). For three rules, the HDE algorithm was faster in 52 instances (90%) and slower in 6 instances (10%). However, the HDE algorithm was not consistently faster than the MDE algorithm for all instances. All cases where HDE was slower occurred in the Scholl-297 problems due to the use of three local search methods and a high number of workstations (ranging from 21 to 50).

After developing the HDE algorithm to minimize the cycle time and testing it on problems ranging from 8 to 297 tasks, its performance was thoroughly evaluated by comparing the obtained results with those of a mathematical model, the GA, and the rule-based heuristics. The comparative analysis demonstrates that the HDE algorithm is capable of finding an improved number of optimal or near-optimal solutions in a shorter computational time compared to the MDE. In addition, the results highlight the robustness and efficiency of the HDE algorithm in addressing both medium-scale and large-scale problems. Specifically, the algorithm consistently outperforms the rule-based heuristics, as illustrated in Figure 6.

TABLE III. COMPARISON OF RULE-BASED HEURISTICS, MDE, AND HDE RESULTS

Example [33]	*m*	Rule-based heuristic [15]				MDE [14]		HDE	
		Two Rules		Three Rules					
		OPT	CT	OPT	CT	OPT	CT	OPT	CT
Mitchell-21	4	27	0.24	27	0.26	27	0.02	27	0.02
	5	22	0.30	21	0.10	21	0.03	21	0.02
	6	18	0.31	18	0.37	18	0.03	18	0.03
Rosenberg-25	4	33	0.22	33	0.27	32	0.06	32	0.03
	5	25	0.14	25	0.11	25	0.03	25	0.02
	6	23	0.19	21	0.43	21	0.03	21	0.03
	7	18	0.35	18	0.43	18	0.03	18	0.03
	8	16	0.43	16	0.50	16	0.04	16	0.03
Heskiaoff-28	6	171	1.16	171	1.27	171	0.03	171	0.03
	7	152	0.29	147	1.18	147	0.03	147	0.03
	8	129	0.82	129	0.82	129	0.04	128	0.03
	9	117	0.63	116	0.58	116	0.05	114	0.03
	10	109	0.27	108	0.77	108	0.06	108	0.06
Buxey-29	5	69	0.23	66	0.50	65	0.02	65	0.03
	6	56	0.34	55	0.60	54	0.03	54	0.03
	7	50	0.24	48	0.55	47	0.03	47	0.03
	8	44	0.25	41	0.59	41	0.03	41	0.03
	9	39	0.16	38	0.45	38	0.05	36	0.03
	10	35	0.29	34	0.57	33	0.03	33	0.05
Sawyer-30	5	65	0.79	65	0.70	65	0.03	65	0.03
	6	55	0.80	54	0.17	54	0.03	54	0.03
	7	48	0.82	47	0.73	47	0.04	47	0.05
	8	41	0.71	41	0.84	41	0.05	41	0.04
Lutz1-32	6	2448	0.22	2404	0.49	2424*	0.03	2404	0.03
	7	2092	0.30	2092	0.37	2092	0.04	2092	0.03
	8	1830	0.18	1816	0.38	1816	0.05	1816	0.05
	9	1644	0.27	1622	0.16	1624*	0.05	1622	0.05
	10	1474	0.30	1474	0.31	1474	0.06	1474	0.05
Gunther-35	5	98	0.97	97	1.09	97	0.05	97	0.03
	6	83	0.56	82	0.73	81	0.05	81	0.05
	7	71	0.65	70	0.75	69	0.05	69	0.05
	8	62	0.70	61	0.82	61	0.05	61	0.05
Kilbridge&Wester-45	6	95	0.61	92	0.41	92	0.08	92	0.07
	7	83	0.88	79	1.90	79	0.10	79	0.09
	8	73	0.89	69	0.39	69	0.11	69	0.10
	9	66	0.77	62	1.61	62	0.12	62	0.11
	10	59	0.89	56	1.77	56	0.14	56	0.13
Hahm-53	5	2948	0.23	2823	1.57	2820	0.09	2806	0.09
	6	2420	0.36	2416	1.21	2416	0.12	2416	0.11
	7	2082	0.51	2013	1.71	2010	0.14	2004	0.14
	8	1840	0.44	1827	1.21	1775	0.16	1775	0.16
	9	2193	0.11	1775	0.15	1775	0.18	1775	0.18
Warnecke-58	10	162	0.92	155	2.66	155	0.24	155	0.24
	11	149	0.96	143	2.59	143	0.26	143	0.25
	12	136	0.76	131	2.29	131	0.28	131	0.28
	13	127	0.95	122	2.15	120	0.31	120	0.31
	14	115	1.27	112	2.14	111	0.33	111	0.33
	15	110	0.88	105	2.50	104	0.35	104	0.36
	16	103	1.03	98	2.10	97	0.37	97	0.37
	17	96	1.34	94	1.83	92	0.40	92	0.40
Tonge-70	10	359	1.54	352	3.87	359**	0.39	351	0.39
	11	323	2.03	320	3.28	320	0.44	320	0.44
	12	298	2.19	295	2.90	293	0.46	293	0.46
	13	275	1.14	273	2.15	275**	0.54	273	0.53
	14	255	1.45	252	2.46	255**	0.55	251	0.57
	15	237	1.26	236	2.00	237**	0.60	236	0.61
	16	232	1.24	222	1.52	232**	0.61	220	0.64
Average computational time			0.68		1.16		0.15		0.15

Remarks: * indicates a worse cycle time than the Three Rules method but better than the Two Rules method, ** indicates worse than the Three Rules method but equal to the Two Rules method, _ indicates the same as the Three Rules method but better than the Two Rules method, and values without asterisks or underscores indicate cycle times equal to or better than both rule-based heuristics.

TABLE IV. COMPARISON OF RULE-BASED HEURISTICS, MDE, AND HDE RESULTS FOR LARGE-SCALE PROBLEMS

Example [33]	c	m	Rule-based heuristic [15]				MDE [14]		HDE	
			Two Rules		Three Rules					
			OPT	CT	OPT	CT	OPT	CT	OPT	CT
Lutz2-89	38	13	40	1.91	38	2.88	38	0.95	38	0.93
	35	14	37	1.92	35	2.89	35	1.03	35	1.01
	33	15	33	2.74	33	2.78	33	1.00	33	1.08
	31	16	31	2.57	31	2.62	31	1.17	31	1.16
	29	17	30	2.38	30	2.45	29	1.24	29	1.22
	27	18	29	1.72	27	2.73	27	1.32	27	1.28
	26	19	27	2.42	27	2.32	26	1.40	26	1.30
	25	20	27	1.71	25	2.65	25	1.47	25	1.41
	24	21	24	2.59	24	2.76	24	1.56	24	1.52
Mukherjee-94	301	14	307	3.67	302	8.12	307**	0.66	302	0.64
	281	15	284	6.12	283	6.92	283	0.69	281	0.69
	263	16	267	3.65	265	6.31	265	0.72	263	0.75
	248	17	252	3.01	251	5.71	251	0.79	248	0.79
	234	18	237	4.55	236	6.05	236	0.85	234	0.81
	222	19	224	4.03	224	4.44	224	0.90	224	0.76
	211	20	223	2.85	212	5.37	221*	0.95	212	0.82
	201	21	211	4.47	205	8.07	210*	0.99	201	0.84
	192	22	203	2.04	193	4.16	201*	1.03	192	1.00
	183	23	193	1.64	185	7.54	193**	1.09	183	1.01
	176	24	186	2.47	179	7.24	186**	1.14	176	1.14
Arcus2-111	16711	9	17208	34.03	16718	31.38	17208**	1.09	16718	1.30
	11570	13	11812	28.96	11576	21.01	11576	1.60	11576	1.91
	10743	14	10776	40.12	10775	16.58	10775	1.78	10775	1.04
	10027	15	10423	12.92	10050	17.15	10223*	1.94	10050	1.13
	9400	16	9836	7.72	9418	16.00	9536*	2.06	9414	1.02
	8847	17	8955	25.85	8862	15.58	8915*	2.17	8862	1.26
	8356	18	8650	12.29	8377	14.58	8650**	2.32	8377	1.34
	7916	19	8143	15.66	7955	10.66	8123*	2.49	7950	1.45
	7520	20	7802	9.99	7558	11.96	7558	2.55	7558	1.51
	7162	21	7388	11.87	7211	9.55	7388**	2.75	7210	1.67
	6837	22	6937	18.23	6861	9.83	6897*	2.93	6859	1.76
	6540	23	6663	15.98	6590	8.24	6620*	3.04	6590	1.85
	6267	24	6340	17.40	6335	8.06	6340**	3.18	6335	1.90
	6016	25	6330	2.53	6086	5.47	6289*	3.31	6066	1.65
Bartholdi-148	805	7	806	19.04	805	20.77	805	2.19	805	2.14
	705	8	712	12.00	705	20.23	705	2.51	705	2.51
	626	9	628	16.38	626	11.33	626	2.85	626	2.86
	564	10	564	14.30	564	15.87	564	3.14	564	3.14
	513	11	520	10.27	513	13.68	513	3.46	513	3.39
	470	12	475	10.03	470	10.51	470	3.79	470	3.79
	434	13	437	11.12	434	24.60	434	4.10	434	3.48
	403	14	405	12.58	403	22.35	403	4.46	403	3.59
Scholl-297	3317	21	3490	10.13	3328	35.65	3460*	20.10	3320	18.98
	3029	23	3189	10.27	3034	53.13	3133*	21.65	3032	19.95
	2787	25	2931	11.32	2790	47.36	2925*	22.57	2790	19.88
	2580	27	2695	25.45	2582	69.36	2684*	24.26	2582	31.22
	2402	29	2526	14.11	2406	48.90	2519*	25.46	2402	33.00
	2247	31	2366	10.67	2256	43.26	2358*	25.25	2252	38.23
	2111	33	2203	16.08	2114	56.53	2144*	27.55	2114	40.15
	1991	35	2011	41.62	1993	62.17	2001*	28.10	1993	40.13
	1883	37	1983	13.43	1888	45.60	1980*	29.22	1885	45.66
	1787	39	1882	14.91	1792	40.66	1880*	31.02	1790	48.07
	1699	41	1779	17.18	1702	58.57	1774*	32.22	1702	47.50
	1620	43	1706	13.87	1623	36.72	1697*	33.12	1623	50.42
	1548	45	1620	12.98	1551	53.30	1616*	37.55	1550	50.14
	1483	47	1562	8.67	1488	42.92	1544*	38.45	1485	55.89
	1422	49	1496	15.44	1438	34.72	1479*	39.25	1438	61.23
	1394	50	1458	12.13	1398	40.51	1421*	40.13	1398	48.12
Average computational time				11.48		20.67		9.60		12.32

Remarks: * indicates a worse cycle time than the Three Rules method but better than the Two Rules method, ** indicates worse than the Three Rules method but equal to the Two Rules method, _ indicates the same as the Three Rules method but better than the Two Rules method, and values without asterisks or underscores indicate cycle times equal to or better than both rule-based heuristics.

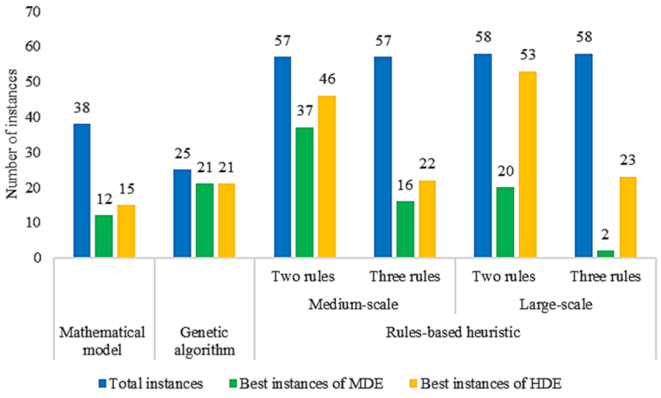

Fig. 6. Comparison of the best results across all heuristics.

V. CONCLUSIONS

This article presents a Hybrid Differential Evolution Algorithm with Local Search (HDE), developed as an enhancement of the Modified Differential Evolution (MDE) algorithm for solving the U-Shaped Assembly Line Balancing Problem Type 2 (UALBP-2). The performance of the HDE algorithm was evaluated by comparing the optimal solutions it generated with those obtained from the mathematical model solved using Lingo V.11, the Genetic Algorithm (GA), and a rule-based heuristic.

The results demonstrate the effectiveness and robustness of the HDE algorithm. It consistently achieves optimal cycle times across all benchmark instances and outperformed the mathematical model in 40% of cases, the GA in 84% of cases, and the rule-based heuristic in approximately 40% of medium- and large-scale problems. While the computational times for the HDE and MDE algorithms were generally similar, the HDE algorithm delivered higher-quality solutions in over 30% of the instances when compared with all other heuristics. These findings highlight the algorithm's ability to efficiently balance computational effort with solution quality.

Despite these promising results, the HDE algorithm may require longer computational times for very large-scale problems. Additionally, the current model does not account for practical factors such as worker fatigue or task variability. Nonetheless, the HDE algorithm demonstrates significant potential for industrial applications, particularly in assembly systems where minimizing cycle time is critical.

Future research could explore several extensions to further enhance the practical applicability of the HDE algorithm. These include real-time implementation and integration with AI-based decision-support systems for enhanced flexibility in Industry 4.0 and 5.0 environments.

ACKNOWLEDGMENTS

We sincerely thank the Department of Industrial Engineering, Faculty of Engineering, Ubon Ratchathani University, for providing access to the Lingo V.11 software. W

APPENDIX

ABBREVIATIONS AND NOTATIONS USED IN THIS ARTICLE

Abbreviation	Meaning
ALB	Assembly Line Balancing
CT	Computational Time
CR	Crossover Rate
DE	Differential Evolution
F	Scaling factor
GALBP	General Assembly Line Balancing Problem
GA	Genetic Algorithm
HDE	Hybrid Differential Evolution Algorithm with Local Search
IoT	Internet of Things
JIT	Just-In-Time
MDE	Modified Differential Evolution
m	Number of workstations
NP	Number of Population
OPT	Optimal cycle time
RAM	Random Access Memory
SALBP-1	Simple Assembly Line Balancing Problem Type 1
UALBP	U-shaped Assembly Line Balancing Problem
UALBP-1	U-Shaped Assembly Line Balancing Problem Type 1
UALBP-2	U-Shaped Assembly Line Balancing Problem Type 2
UALBP-E	U-Shaped Assembly Line Balancing Problem Type E

REFERENCES

[1] Z. Li, M. Janardhanan, Q. Tang, and Z. Zhang, "Models and algorithms for U-shaped assembly line balancing problem with collaborative robots," *Soft Computing*, vol. 27, no. 14, pp. 9639–9659, Jul. 2023, https://doi.org/10.1007/s00500-023-08130-y.

[2] A. Farsi, M. Mokhtarzadeh, M. Rabbani, N. Manavizadeh, and M. Ghasempour Anaraki, "Using parallel metaheuristics to solve a parallel U-shaped robotic mixed-model assembly line balancing and sequencing problem," *Soft Computing*, vol. 28, no. 21, pp. 12603–12621, Nov. 2024, https://doi.org/10.1007/s00500-024-10311-2.

[3] Y. Karatepe Mumcu, "Solution approach using heuristic and artificial neural networks methods in assembly line balancing problems: A case study in the lighting industry," *Heliyon*, vol. 10, no. 5, Mar. 2024, Art. no. e26950, https://doi.org/10.1016/j.heliyon.2024.e26950.

[4] S. Chulakit *et al.*, "A Centralized IOT-Based Process Cycle Time Monitoring System for Line Balancing Study," *Journal of Advanced Research in Applied Mechanics*, vol. 105, no. 1, pp. 58–67, Jun. 2023, https://doi.org/10.37934/aram.105.1.5867.

[5] G. J. Miltenburg and J. Wijngaard, "The U-line Line Balancing Problem," *Management Science*, vol. 40, no. 10, pp. 1378–1388, Oct. 1994, https://doi.org/10.1287/mnsc.40.10.1378.

[6] N. Kriengkorakot and N. Pianthong, "The Assembly Line Balancing Problem : Review articles," *Engineering and Applied Science Research*, vol. 34, no. 2, pp. 133–140, Jun. 2007.

[7] 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, Mar. 2020, Art. no. 21, https://doi.org/10.3390/joitmc6010021.

[8] N. Kriengkorakot and N. Pianthong, "The U-line Assembly Line Balancing Problem," *KKU Engineering Journal*, vol. 34, no. 3, pp. 267–274, Jun. 2007.

[9] M. Fathi, M. J. Álvarez, and V. Rodríguez, "A new heuristic-based bi-objective simulated annealing method for U-shaped assembly line balancing," *European Journal of Industrial Engineering*, vol. 10, no. 2, pp. 145–169, Apr. 2016, https://doi.org/10.1504/EJIE.2016.075849.

[10] Z. Li, M. N. Janardhanan, and H. F. Rahman, "Enhanced beam search heuristic for U-shaped assembly line balancing problems," *Engineering*

Optimization, vol. 53, no. 4, pp. 594–608, Apr. 2021, https://doi.org/10.1080/0305215X.2020.1741569.

[11] 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, Aug. 2008, https://doi.org/10.1080/00207540701247906.

[12] V. Jonnalagedda and B. Dabade, "Application of Simple Genetic Algorithm to U-Shaped Assembly Line Balancing Problem of Type II," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 6168–6173, Jan. 2014, https://doi.org/10.3182/20140824-6-ZA-1003.01769.

[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, Dec. 2018, Art. no. 79, https://doi.org/10.3390/mca23040079.

[14] K. Chantarasamai and O.-U. Lasunon, "Modified Differential Evolution Algorithm for U-Shaped Assembly Line Balancing Type 2," *International Journal of Intelligent Engineering and Systems*, vol. 14, no. 4, pp. 452–462, Aug. 2021, https://doi.org/10.22266/ijies2021.0831.39.

[15] 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, Aug. 2017, https://doi.org/10.1016/j.apm.2016.12.031.

[16] 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, Mar. 1999, https://doi.org/10.1080/002075499191481.

[17] N. Kriengkorakot and P. Kriengkorakot, "Heuristics comparison for u-shaped assembly line balancing in the apparel factory," *KKU Engineering Journal*, vol. 41, no. 2, pp. 155–162, Jun. 2014, https://doi.org/10.14456/kkuenj.2014.3.

[18] Ö. F. Yılmaz, "Robust optimization for U-shaped assembly line worker assignment and balancing problem with uncertain task times," *Croatian Operational Research Review*, vol. 11, no. 2, pp. 229–239, Dec. 2020, https://doi.org/10.17535/crorr.2020.0018.

[19] R. Storn and K. Price, "Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec. 1997, https://doi.org/10.1023/A:1008202821328.

[20] X. Wu, X. Liu, and N. Zhao, "An improved differential evolution algorithm for solving a distributed assembly flexible job shop scheduling problem," *Memetic Computing*, vol. 11, no. 4, pp. 335–355, Dec. 2019, https://doi.org/10.1007/s12293-018-00278-7.

[21] P. Sriboonchandr, N. Kriengkorakot, and P. Kriengkorakot, "Improved Differential Evolution Algorithm for Flexible Job Shop Scheduling Problems," *Mathematical and Computational Applications*, vol. 24, no. 3, Sep. 2019, Art. no. 80, https://doi.org/10.3390/mca24030080.

[22] X. Li *et al.*, "A Q-learning improved differential evolution algorithm for human-centric dynamic distributed flexible job shop scheduling problem," *Journal of Manufacturing Systems*, vol. 80, pp. 794–823, Jun. 2025, https://doi.org/10.1016/j.jmsy.2025.04.001.

[23] 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, Dec. 2018, Art. no. 55, https://doi.org/10.3390/mca23040055.

[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, Sep. 2018, Art. no. 40, https://doi.org/10.3390/mca23030040.

[25] R. Kamphukaew, K. Sethanan, T. Jamrus, and H. 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, Dec. 2014, https://doi.org/10.14456/easr.2018.37.

[26] 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, Sep. 2018, Art. no. 34, https://doi.org/10.3390/mca23030034.

[27] R. V. V. Krishna and S. S. Kumar, "Hybridizing Differential Evolution with a Genetic Algorithm for Color Image Segmentation," *Engineering, Technology & Applied Science Research*, vol. 6, no. 5, pp. 1182–1186, Oct. 2016, https://doi.org/10.48084/etasr.799.

[28] A. C. Nearchou, "A Differential Evolution Algorithm for Simple Assembly Line Balancing," *IFAC Proceedings Volumes*, vol. 38, no. 1, pp. 247–252, Jan. 2005, https://doi.org/10.3182/20050703-6-CZ-1902.01463.

[29] 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, Feb. 2015, https://doi.org/10.1080/21681015.2015.1007094.

[30] A. C. Nearchou and S. L. Omirou, "Assembly Line Balancing Using Differential Evolution Models," *Cybernetics and Systems*, vol. 48, no. 5, pp. 436–458, Jul. 2017, https://doi.org/10.1080/01969722.2017.1319238.

[31] R. Pitakaso and K. Sethanan, "Modified differential evolution algorithm for simple assembly line balancing with a limit on the number of machine types," *Engineering Optimization*, vol. 48, no. 2, pp. 253–271, Feb. 2016, https://doi.org/10.1080/0305215X.2015.1005082.

[32] M. Zakaraia, H. Zaher, and N. Ragaa, "Solving stochastic multi-manned U-shaped assembly line balancing problem using differential evolution algorithm," *International Journal of Production Management and Engineering*, vol. 10, no. 1, pp. 13–22, Jan. 2022, https://doi.org/10.4995/ijpme.2021.16084.

[33] A. Scholl, "Benchmark Data Sets by Scholl (1993)." Assembly Line Balancing, 1993. [Online]. Available: https://assembly-line-balancing.de/salbp/benchmark-data-sets-1993/.

AUTHORS PROFILE

Krit Chantarasamai is an Assistant Professor at the Rajamangala University of Technology Isan, Surin Campus, Thailand. He obtained his Ph.D. in Mechanical Engineering from Mahasarakham University, Thailand, and received both his B.E. and M.E. degrees in Industrial Engineering from Ubon Ratchathani University, Thailand. Over the past decade, he has gained extensive experience in academic and non-academic fields, actively contributing to various research and academic projects. He has also published several research articles in reputable international journals.

Visit Junchuan is a Lecturer at Rajamangala University of Technology Isan, Surin Campus, Thailand. He holds a Ph.D., an M.E., and a B.E. in Mechanical Engineering from Ubon Ratchathani University, Thailand. He has acquired both academic and non-academic experience through participation in various research and academic projects, and has published several research articles in reputable international journals.

Poontana Sresracoo is an Assistant Professor at Buriram Rajabhat University, Thailand. He obtained his Ph.D. in Industrial Engineering from Ubon Ratchathani University, Thailand. He received his M.E. degree in Industrial Engineering from Chiang Mai University, Thailand, and his B.E. degree in Industrial Engineering from Ubon Ratchathani University, Thailand. Over the past decade, he has actively participated in various research and academic projects and has published research articles in reputable international journals.