International Journal of Recent Engineering Research and Development (IJRERD)
ISSN: 2455-8761
www.ijrerd.com || Volume 07 – Issue 05 || May 2022 || PP. 01-13

Cat Swarm Optimization Algorithm for Solving Multi-Objective Sequence-Dependent U-Shaped Disassembly Line Balancing Problem

Pengfei Yao[1], Surendra M. Gupta[2]
[1]*Department of Mechanical and Industrial Engineering*
Northeastern University Boston, MA, 02115, USA
[2]*Department of Mechanical and Industrial Engineering*
Northeastern University Boston, MA, 02115, USA

Abstract: Rapidly updating technology and accelerating development in manufacturing greatly improve quality of life for all individuals. For example, electronic products are updated frequently. Obvious consequences of these trends are shortened lifecycle and increase in end-of-life (EOL) products. This necessitates governments to enactlaws and regulations addressing environmental consciousness towards product manufacturing and consumption. In contrast to traditional landfilling of EOL products, product recovery is regarded as efficient, eco-friendly and effective option. Remanufacturing, reusing and recycling are three popular options of product recovery. In remanufacturing, disassembly is often the first and one of the most important steps. Aiming to physically separate EOL products into parts or subassemblies, disassembly is operated on a paced line linked with different workstations. Therefore, balancing the disassembly line is extremely important for efficiency and profitability. U-shaped disassembly line is different from the traditional straight-line layout because operators or robotic machines can work across stations. This line characteristic improves efficiency and profitability. Sequence-dependency is an issue that needs to be addressed as well. Whenever a task interacts with another task, their task times may be influenced due to change in the sequence in which tasks are performed. Multiple objectives are considered in this paper, viz., number of workstations, line smoothness, hazardous parts removal, and highly demanded parts removal. Disassembly line balancing problem (DLBP) belongs to the NP-hard class which makes it difficult to optimize using traditional mathematical programming methods for practical problems. For this reason, this paper proposes the use of a novel meta-heuristic algorithm called cat swarm optimization algorithm (CSO). Real instances are presented in this paper to test the ability of CSO algorithm and to compare its performance between U-shaped and straight-line disassembly lines. Results show that CSO has a great searching ability in finding near-optimal solutions and U-shaped layout improves line efficiency in many aspects. Comparative study is used to evaluate the performance of CSO with several other meta-heuristic algorithms. Benchmark instances are used in the comparative study. Results demonstrate that CSO outperforms many algorithms and is superior in minimizing the number of workstations.

Keywords: Remanufacturing, Disassembly line balancing problem (DLBP), Sequence-dependent U-shaped disassembly line, Cat swarm optimization algorithm (CSO).

1. Introduction

Development of technology and manufacturing skills results in diversification of products. Gradual changes in consumptions habits of individuals and fast updated products especially electronic products have resulted in shorten lifespan of products. An increasing number of waste or end-of-life (EOL) productsis rising attentions of individuals and society. (Yin et al., 2022). Waste problem is a global challenge to all countries and individuals, and it is extremely harmful to healthy and environment especially electronic and electrical waste problem (Li and Janardhanan, 2021). One of the most popular and straight optimal dealing with waste problem is landfilling. However, invisible threaten is still affecting individuals and environment like soil pollution will directly affect our food safety. Considering of increasing pressure of environment and stricter regulations made by governments, green concept should be added to the processes of dealing with EOL products (Akpinar, Ilgin, and Aktas, 2021; Chen et al., 2021). In contrast to landfilling, product recovery aims to protect the environment and receive part of value of EOL products (Edis, Edis, and Ilgin, 2021; Gao et al., 2021; Yao and Gupta, 2021f). A more basic concept, environmentally conscious manufacturing and product recovery was first highlighted by Gungor and Gupta (1999b) with the considering of adding green concept to the whole lifecycle of a product. There are three familiar options in product recovery, viz., remanufacturing, reuse, and recovery. Among these three options, disassembly is often the first and one of the most important steps. Disassembly is much more complex than assembly due to the different operations and objectives. It aims to physically separate EOL

products into parts or subassemblies via a paced line linked with workstations. Considering of improvement of line efficiency and profitability, U-shaped disassembly line is applied in this paper. Except of traditional straight-line layout, parallel and two-sided lines are two other types of disassembly line. A highly efficient disassembly line plays an important role in today's modern industrial environment;therefore, the topic of balancing disassembly lines attracts academics and practitioners (Li, Kucukkoc, and Zhang, 2019).

Disassembly line balancing problem (DLBP) was first proposed by Gungor and Gupta (1999), it aims at optimally assign disassembly tasks to workstations within the domain of precedence relationships and cycle time constrains. DLBP is not just the reverse loop of assembly line balancing problem (ALBP), since the special line operations made DLBP much more complicated than ALBP. There are much complex precedence relationships in DLBP, viz., AND precedence, OR precedence, and complex AND/OR precedence.

Since the pioneering work published by Gungor and Gupta (1999a), DLBP has becoming an active research area and methods and approaches are continually applied on DLBP. The nature of DLBP is optimization problem and it belongs to NP-hard problem which is proven by McGovern and Gupta (2007a, 2007b). One obvious characteristic of NP-hard problem is for large-size instances, exact methods and approaches cannot find optimal solutions in a reasonable computational time, for this reason, heuristic and meta-heuristic algorithms can help to find near-optimal solutions in a short time. According to the information in Ozceylan et al. (2019) and published papers after that, there are many suitable and ability-proved methods and approaches. These include genetic algorithm (McGovern and Gupta, 2007; Kalayci, Polat, and Gupta, 2016), artificial bee colony optimization (Kalayci and Gupta, 2013a; Wang, Guo, and Liu, 2019; Li et al., 2021), ant colony optimization (Agrawal and Tiwari, 2008; Zhu et al., 2014; Yao and Gupta, 2021c), cat swarm optimization (Yao and Gupta, 2021a), tabu search (Kalayci and Gupta, 2014), particle swarm optimization (Kalayci and Gupta, 2013b), small world optimization (Yao and Gupta, 2021b), simulated annealing algorithm (Kalayci and Gupta, 2013c; Wang et al., 2021), Invasive weed optimization (Yao and Gupta, 2021d), teaching-learning based optimization (Yao and Gupta, 2021e), fish school search optimization (Yao and Gupta, 2021f), and artificial fish swarm algorithm (Zhang et al., 2017).

The rest of the paper is structured as follows: literature review is included in the second section. The section that follows presents a mixed-integer non-linear programming(MINLP) model with four objectives and constraints, and detailed DLBP description is included. This is followed by a section that covers case study and comparative study. The last section includes conclusion and directions for future research.

2. Literature review

Balancing disassembly line plays critical role in solving DLBP. There are four typical types of a disassembly line which are straight-line, parallel line, U-shaped line, and two-sided line. Most of the published studies applied their approaches on a straight-line disassembly line (Ozceylan et al., 2019) and only a few papers considered U-shaped line, which is more complex than straight-line layout.U-shaped disassembly line was first used in Agrawal and Tiwari (2008) and from that time, U-shaped DLBP is getting more attentions. Avikal and Mishra (2012) and Avikal, Jain, and Mishra (2013) applied two different heuristic algorithms on a U-shaped layout. Zhang et al. (2018) combined genetic algorithm and pareto hybrid ant colony optimization together and applied on a U-shaped disassembly with the consideration of multiple objectives. A special type of DLBP, Sequence-dependent U-shaped DLBP was first researched in Li, kucukkoc, and Zhang (2019) and authors introduced a novel iterated local search strategy to compare performances of different algorithms.Another special type of UDLBP, partial UDLBP has been first introduced by Wang, Gao, and Li (2020) and Li and Janardhanan (2021). Yao and Gupta (2021a) and Yao and Gupta (2021b) have, for the first-time, introduced cat swarm optimization (CSO) and small world optimization (SWO) on a U-shaped layout with the considering of multiple objectives respectively. Wang et al. (2021) applied complex and combined algorithms to help balance a U-shaped disassembly line. Yao and Gupta (2021c, 2021d) first presented ant colony optimization (ACO) and invasive weed optimization (IWO) on a U-shaped line and multiple-objective complete disassembly was considered. A two-phase artificial bee colony algorithm andan original bee algorithm were proposed and implemented on a U-shaped line (Li et al., 2021). Considering of stochastic characteristic of DLBP, Xu et al. (2021) introduced a combined strategy on a U-shaped line. Multiple products partial DLBP was first considered by Wang et al. (2021) and results show that U-shaped disassembly line improves line efficiency greatly. Most recently, Yao and Gupta (2021e) and Yao and Gupta (2021f) first implemented and tested teaching-learning-based optimization (TLBO) and fish school search algorithm (FSS) on a U-shaped line respectively and multiple objectives are considered to fulfill real world demands. U-shaped disassembly line has one advantage which allows operators or robotic machines work across the workstation. This line type gives more allocations of tasks which may improve line efficiency and decrease the chance of line failure.

Sequence-dependent relationship is a special precedence relationship, and many industrial cases need to consider it. Different task removal sequence may affect the task processing time, therefore in this paper, sequence dependency is taking into account.

As mentioned above, DLBP belongs to NP-hard class problem, therefore meta-heuristics and heuristic algorithms are introduced on disassembly line continually. Cat swarm algorithm (CSO) was first proposed by Chu, Tsai, and Pan (2006) and it has for the first time applied on DLBP in research (Yao and Gupta, 2021a). CSO is a novel swarm-based meta-heuristic algorithm and is generated by observing the behaviors of cats. Two important factors in CSO are tracking and seeking mode. The mechanism of these two modes are local search and global search, and according to the study of Chu, Tsai, and Pan (2006) and Yao and Gupta (2021a), CSO has a great ability to solve optimization problem and it shows a superior searching ability on a U-shaped disassembly line. The main contributions of this paper are presented as follows:

(1) Since one of the objectives is minimization total of idle times is a non-linear function, therefore a mixed-integer non-linear programming (MINLP) model is introduced to help optimize four different objectives.
(2) Sequence-dependent U-shaped DLBP (SUDLBP) is studied and CSO has for the first time, applied to optimize this type of DLBP.
(3) Two sets of instances are implemented in this paper to test the performance of CSO and to compare U-shaped line and straight-line disassembly line. The first set contains two small-size instances, and the computational tests show that the U-shaped layout obtains a better performance than the traditional straight-line layout especially considering minimization number of workstations and minimization total idle times. The second instance set contains 47 benchmark problem, and the comparative study presents that CSO has a strong searching ability compare with other meta-heuristic algorithms. Total of 10 algorithms are compared include hill-climbing algorithm (HC) (McGovern and Gupta, 2007a, 2007b), late acceptance hill-climbing algorithm (LAHC) (Yuan, Zhang, and Shao, 2015), simulated annealing algorithm (SA) (Kalayci and Gupta, 2013c), tabu search algorithm (TS), genetic algorithm (GA), artificial bee colony algorithm (ABC) (Kalayci and Gupta, 2013a), bee algorithm (BA) (Li et al., 2021), particle swarm optimization (PSO) (Kalayci and Gupta, 2013b), and iterated local search optimization (ILS) (Li, Kucukkoc, and Zhang, 2019).

3. Problem definition

This section first introduces SUDLBP and then MINLP model with related notations and constraints are presented.

3.1 Problem statement

The nature of DLBP is optimally allocate disassembly tasks to workstations with the domain of cycle time and precedence relationship constraints. There are four optimization criteria in this paper, viz., minimizing number of workstations, minimizing total of idle times, removing hazardous part(s) early, and removing high demand part(s) early, which are also known as objectives. Cycle time constraints should be strictly followed to avoid line stoppage, it limits that the total task processing times of each workstation should be less than or equal to the predetermined cycle time. Precedence relationship constraint, as mentioned in previous section, indicates that all the AND predecessors or at least one OR predecessor must be allocated before target task. Sequence-dependent relationship is a special type of precedence relationship with adding sequence dependency to tasks with sequence-dependent situation. A small-size instance (Kalayci and Gupta, 2014) is shown in Fig. 1 and Table 1, detailed instance information is presented. In Fig. 1 dashed lines represent linked two tasks have sequence-dependent relationship. The sequence dependencies of this 8-part PC instance are as follows: $sd_{2,3} = 2$, $sd_{3,2} = 4$, $sd_{5,6} = 1$, and $sd_{6,5} = 3$. If task 2 is disassembled before task 3, the task processing time of task 2 should be $t_2 + sd_{3,2} = 14$. Otherwise, if task 3 is removed before task 2, task processing time of task 3 should be $t_3 + sd_{2,3} = 12 + 2 = 14$. Also, it is the same calculation of task 5 and task 6. If task 5 is allocated before task 6, the actual task processing time of task 5 is $t_5 + sd_{6,5} = 23 + 3 = 26$.

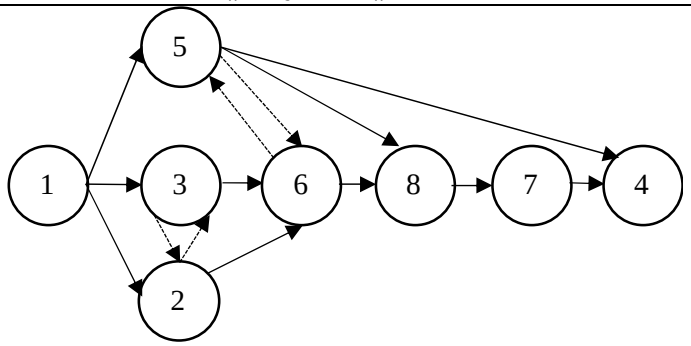

Fig. 1. Precedence relationship of 8-part PC instance

Table 1. Information of the 8-part PC instance

Task number	Part name	Task removal time	Hazardous index	Demand
1	PC top cover	14	No	360
2	Floppy drive	10	No	500
3	Hard drive	12	No	620
4	Back plane	18	No	480
5	PCI cards	23	No	540
6	RAM modules	16	No	750
7	Power supply	20	No	295
8	Motherboard	36	No	720

3.2 Model formulation

In this section, detailed notations, objective formulations, and constrains are presented. Assumptions should be mentioned in this paper:

(1) EOL products are enough and there is only one type of product.
(2) All parts of the product are disassembled (complete disassembly).

Notation

N Number of tasks
i,j Task index, $i, j = 1,2, \dots, N$
M Number of workstations
m Workstation (sub-station) index, $m = 1,2, \dots, 2M$
t_i Processing/removal time of task i
h_i Binary variable, 1, if task i is hazardous; 0, otherwise
d_j Demand value of task j
ANDP(i) Set of AND predecessor of task i
ORP(i) Set of OR predecessor of task i
CT Cycle time
T_m Total task processing times of workstation m
sd_{ij} Sequence dependent time between task j and task i
F_a Objective function, $a = 1,2,3,4$

Decision variables:

x_{im} Binary variable, 1, if task i is assigned to sub-station m; 0, otherwise
x_{imj} Binary variable, 1, if task is assigned to sub-station m and is operated before task j; 0, otherwise
w_{ij} Binary variable, 1, if task i is operated before task j; 0, otherwise
ws_m Binary variable, 1, if workstation m is opened; 0, otherwise
s_i Position number of task i in sequence

On a U-shaped line, one workstation has two sides, entrance side and exit side. To solve the problem conveniently, one workstation is divided into two sub-stations. That means, sub-station $1, 2, \dots, M$ are on the entrance side, sub-station $M + 1, M + 2, \dots, 2M$ are on the exit side.

Objectives:

$$\text{Min } F_1 = \sum_{m=1}^{M} ws_m \quad (1)$$

$$\text{Min } F_2 = \sum_{m=1}^{M} (CT - T_m)^2 \quad (2)$$

$$\text{Min } F_3 = \sum_{i=1}^{N} (s_i * h_i) \quad (3)$$

$$\text{Min } F_4 = \sum_{i=1}^{N} (s_i * d_i) \quad (4)$$

The first objective is to minimize the number of workstations. Equation (2) is a non-linear objective with the goal of optimally balancing line smoothness. Equation (3) presents removal of hazardous part(s) early and equation (4) tries to remove high demand part(s) early.

Constraints:

$$\sum_{m=1}^{M} (x_{im} + x_{i,2M+1-m}) = 1 \ \forall i \quad (5)$$

$$\sum_{i=1}^{N} (x_{im} + x_{i,2M+1-m}) \geq 1 \ \forall i \quad (6)$$

$$CT \geq T_m \ \forall m \quad (7)$$

$$x_{im} \leq \sum_{n=1}^{m} x_{jn} \ \forall i, m; \ \forall j \in ANDP(i) \quad (8)$$

$$x_{im} \leq \sum_{j \in ORP(i)} \sum_{n=1}^{m} x_{jn} \ \forall m, \forall i \in ORPT \quad (9)$$

$$w_{ij} + w_{ji} = 1 \forall i, j \ and \ i < j \quad (10)$$

$$T_m = \sum_{i=1}^{N} t_i \times (x_{im} + x_{i,2M+1-m}) + \sum_{i=1}^{N} \sum_{j=1}^{N} sd_{ji} \times (x'_{imj} + x'_{i,2M+1-m,j}) \ \forall i, j \quad (11)$$

$$s_i = N - \sum_{j=1}^{N} w_{ij} \ \forall i \quad (12)$$

Constraint (5) ensures that one task can only be allocated to one sub-station. Constraint (6) indicates that one workstation can operate one or more tasks. Cycle time constraint is shown in constraint (7). Constraint (8) and (9) are AND type precedence relationship and OR type precedence relationship. As mentioned in previous section, AND predecessors of a task should be removed at previous or the same sub-station. For OR predecessors of a task, at least one of its OR predecessor should be assigned at previous or the same sub-station. Constraint (10) ensures that one task is either removed before another task or after it. Constraint (11) and (12) introduce the calculation of total task processing time of a workstation and sequence location of a task.

For multi-objective optimization problem, there are two popular ways to classify near-optimal solutions, namely, hierarchy method and pareto optimal method. Hierarchy method sets priority to objectives and pareto optimal method do not set weight to objectives. In research of DLBP field, hierarchy method is applied in much research like McGovern and Gupta (2006), Kalayci and Gupta (2013a, 2013b, 2013c, 2014), Li, Kucukkoc, and Zhang (2019), and Li et al. (2021). For pareto optimal method, based on the mechanism of it, many solutions can be chosen, and the set of near-optimal solutions is much larger. Research like Yao and Gupta (2021a, 2021b, 2021c, 2021d, 2021e, 2021f) used this method to classify near-optimal solutions.

4. Cat swarm optimization algorithm (CSO)

Cat swarm optimization (CSO) algorithm is a swarm intelligence (SI) based optimization algorithm which is inspired by the behaviors of cats. This algorithm was first introduced by Chu and Tsai (2006). The hunting ability of cats is observed, therefore, inspired by this, in CSO there are two kinds of mode: seeking mode and tracing mode. In test environment, a population of cats are randomly distributed in the space and are divided into two groups. Cats in the first group are resting which is seeking mode and cats in the second group are moving around chasing a target which is called tracing mode. Solutions are evaluated by objectives and near-optimal results will be found. In this paper, the percentage of seeking and tracing mode cats are 70% and 30% respectively. Based on the study of Chu and Tsai (2006), steps of CSO are showing below:

Step 1: Create the initial population of cats and set them at M-dimensional solution space. Based on the velocity rules, assign a velocity value to each cat.
Step 2: Divide cats into two groups.
Step 3: Evaluate objective value(s) of each cat and store the best solution.
Step 4: Check the termination criteria to decide whether it needs to repeat or stop.
Based on above mentioned steps of CSO, part of it needs to be changed to fulfill objectives of SUDLBP. c_1 and c_2 are the percentage of seeking mode cats and tracing mode cats, respectively.

4.1 Encoding and decoding

Task permutation sometimes is not the same with task sequence on a U-shaped disassembly line. In this paper, the rule of task permutation for encoding is the same with research (Kalayci and Gupta, 2013a, 2013b, 2013c, 2014). A feasible solution of the 8-part PC instance is presented in Fig.2.

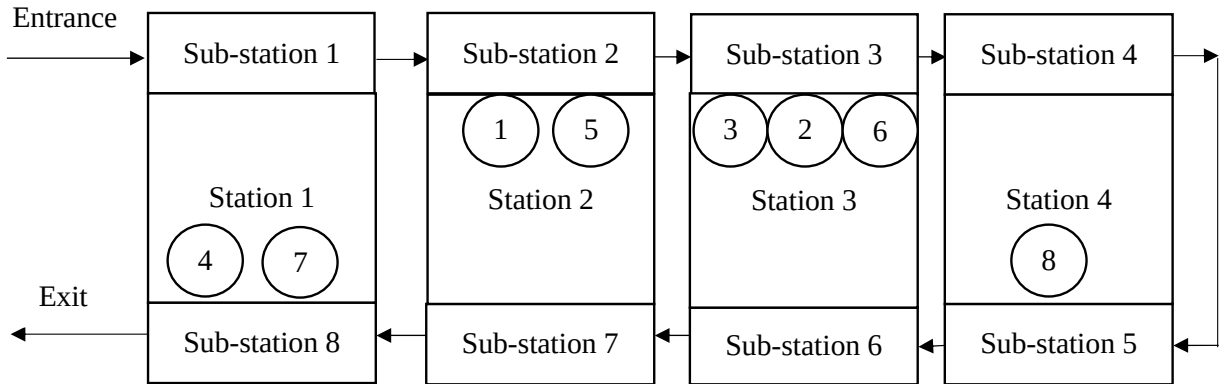

Fig.2. Task assignment of the feasible solution

Task permutation of this feasible solution is 4, 7, 1, 5, 3, 2, 6, 8, but the sequence of this solution is 1, 5, 3, 2, 6, 8, 7, 4. The difference between these should be done by decoding procedure. Decoding procedure aims to transfer encoding into a feasible solution and the decoding procedure of SUDLBP is much more complicated that that of DLBP and SDLBP. The detailed objective values are presented in Table 2.

Algorithm 1. Decoding procedure for SUDLBP

Start
Step 1: If all tasks are assigned, terminate procedure; otherwise, execute step 2.
Step 2: Open a new station.
Step 3: Add task(s), whose predecessor(s) has been assigned to the entrance side, to the available task set A_{en}; Add task(s), whose successor(s) has been assigned to the exit side, to the available task set A_{ex}.
Step 4: Add the task in $A_{en}$to the assignable task set AS_{en} on the entrance side with the domain of cycle time constraint; Add the task in A_{ex} to the assignable task set AS_{ex} on the exit side with the domain of cycle time constraint. % For an assignable task, it can be assigned only the total task processing time of this workstation is less than or equal to the given cycle time with the considering of sequence dependency.
Step 5: If both two assignable task sets AS_{en} and AS_{ex} are empty, go back to step 1; otherwise, execute step 6.
Step 6: Select the task with higher priority of task permutation and allocate it to the entrance or exit side based on the situation; go back to step 3.
End

Table 2. Objective values of the feasible solution

Workstation number	Sub-station	Task number	Task processing time	Total task processing time	Idle time
Workstation 1	Sub-station 1	-	-	38	2
	Sub-station 8	7,4	20,18		
Workstation 2	Sub-station 2	1,5	14,23+3	40	0
	Sub-station 7	-	-		
Workstation 3	Sub-station 3	3,2,6	12+2,10,16	40	0
	Sub-station 6	-	-		
Workstation 4	Sub-station 4	-	-	36	4
	Sub-station 5	8	36		
F_1	4				
F_2	2^2+0+0+4^2=20				
F_3	0				
F_4	$1 \times 360 + 2 \times 540 + 3 \times 620 + 4 \times 500 + 5 \times 750 + 6 \times 720 + 7 \times 295 + 8 \times 480 = 19275$				

The pseudo code of CSO is presented as follows:

Pseudo code of CSO Algorithm

1. Input: Objective function(s), Fitness function(s), N_p, T, w, c_1, c_2.
2. Initialize a random feasible population (P).
3. Evaluate objective functions (F) and fitness functions (Fitness).
4. For t=1 to T
   If $i \leq c_1$

Perform Seeking mode Phase for all cats.
Save the best solution.
else
Perform Tracing mode Phase to generate N_p solutions.
Save the best solution.
End
End

5. Computational study

This section aims to present the searching ability of proposed CSO, and the performance compared with other meta-heuristic algorithms. Two sets of instances are involved: the first set contains two small-size instances which are obtained from research (Kalayci and Gupta, 2013a, 2013b, 2014; Li, Kucukkoc, and Zhang, 2019), and the second set contains 47 benchmark instances. A 10-part product (P10) and a 25-part product (P25) are combined to be small-size instance set. Fig. 3 and Table 3 present detailed information of P10 instance which includes task number, part removal time, hazardous index, and demand values. Section 5.1 and 5.2 present comparation results of U-shaped and straight-line disassembly line and comparation performance of different algorithms respectively.

5.1 Case study

The sequence dependencies of P10 are presented as follows: $sd_{1,4} = 1$, $sd_{4,1} = 4$, $sd_{2,3} = 2$, $sd_{3,2} = 3$, $sd_{4,5} = 4$, $sd_{5,4} = 2$, $sd_{5,6} = 2$, $sd_{6,5} = 4$, $sd_{6,9} = 3$, and $sd_{9,6} = 1$. The proposed CSO algorithm is applied on a straight-line and a U-shaped disassembly line separately with a cycle time of 40. For each line type, CSO run 20 times and parameters for two line types are the same. Table 4 presents detailed comparation results, and the best value, average value, and standard deviation are listed.

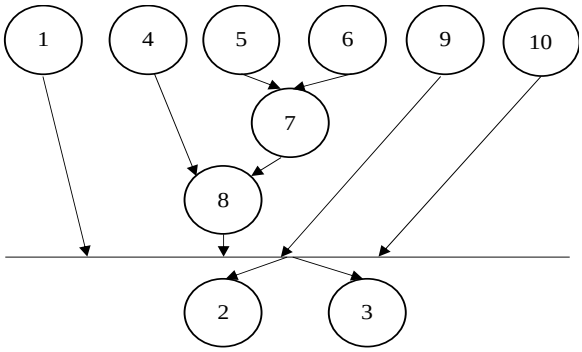

Fig. 3. Precedence relationship diagram of P10

Table 3. Instance information of P10

Task number	Part removal time	Hazardous index	Demand value
1	14	No	0
2	10	No	500
3	12	No	0
4	17	No	0
5	23	No	0
6	14	No	750
7	19	Yes	295
8	36	No	0
9	14	No	360
10	10	No	0

Table 4. Performance of two types of disassembly line on P10

Line type	Algorithm	Evaluation	F_1	F_2	F_3	F_4
Straight-line	SWO	Best value	5	67	5	9605
		Avg. value	5.00	67.00	5.00	9605.00
		SD	0.00	0.00	0.00	0.00
U-shaped	SWO	Best value	5	61	6	8880
		Avg. value	5.00	61.00	6.00	8880.00
		SD	0.00	0.00	0.00	0.00

Based on the results of Table 4, U-shaped layout has a better performance on minimizing total idle times and removing high demand part(s) early. For minimizing number of workstation, two lines found the same best value, which is 5. It is sufficient to conclude that U-shaped disassembly line can improve line efficiency and has higher flexibility than traditional straight-line disassembly line. Fig. 4 and Table 5 introduce detailed information of the second instance (P25), and also this instance is acquired from research (Kalayci and Gupta, 2013a).Sequence dependencies of P25 instance are shown as follows: $sd_{4,5} = 2$, $sd_{5,4} = 1$, $sd_{6,7} = 1$, $sd_{7,6} = 2$, $sd_{6,9} = 2$, $sd_{9,6} = 1$, $sd_{7,8} = 1$, $sd_{8,7} = 2$, $sd_{13,14} = 1$, $sd_{14,13} = 2$, $sd_{14,15} = 2$, $sd_{15,14} = 1$, $sd_{20,21} = 1$, $sd_{21,20} = 2$, $sd_{22,25} = 1$, and $sd_{25,22} = 2$. CSO algorithm is also applied on two types of lines 20 times with a cycle time of 18. The performance of two types of line is presented in Table 6. In Table 6, all objective values of U-shaped line are better or equal to that of straight-line disassembly line. Again, results of P25 show that U-shaped layout has better assignment ability and also, CSO has the ability on small-size instances.

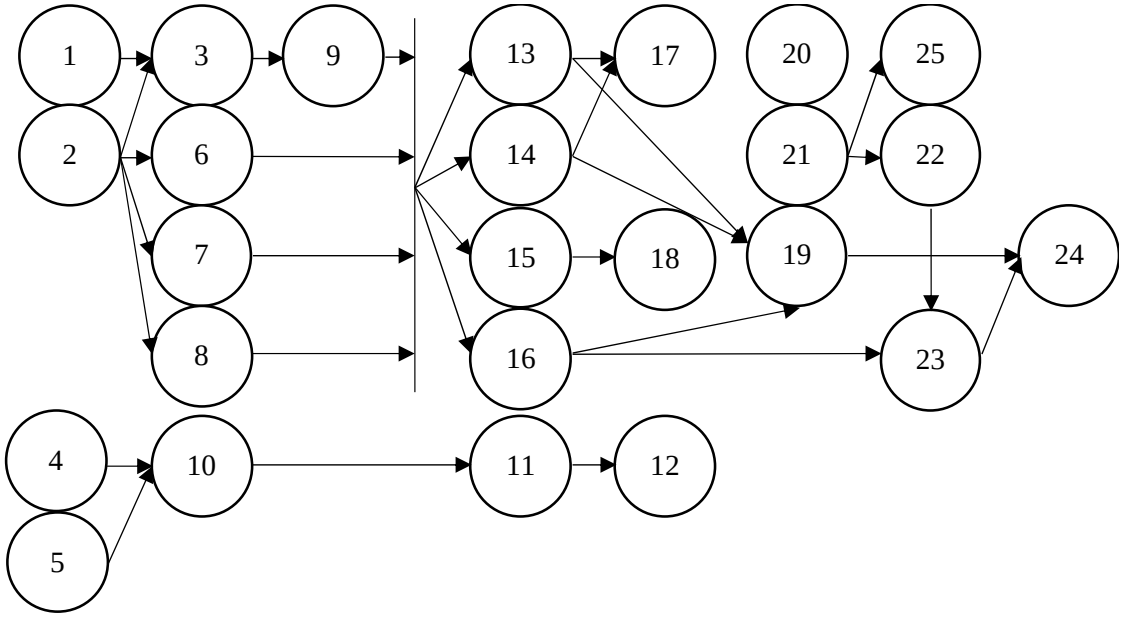

Fig 4. Precedence relationship diagram of P25

Table 5. Instance information of P25

Task number	Part name	Part removal time	Hazardous index	Demand value
1	Antenna	3	1	4
2	Battery	2	1	7
3	Antenna guide	3	0	1
4	Bolt (Type 1) A	10	0	1
5	Bolt (Type 1) B	10	0	1
6	Bolt (Type 2) 1	15	0	1
7	Bolt (Type 2) 2	15	0	1
8	Bolt (Type 2) 3	15	0	1
9	Bolt (Type 2) 4	15	0	1
10	Clip	2	0	2
11	Rubber Seal	2	0	1
12	Speaker	2	1	4
13	White Cable	2	0	1
14	Red/Blue Cable	2	0	1
15	Orange Cable	2	0	1
16	Metal Top	2	0	1
17	Front Cover	2	0	2
18	Back Cover	3	0	2
19	Circuit Board	18	1	8
20	Plastic Screen	5	0	1
21	Keyboard	1	0	4
22	LCD	5	0	6
23	Sub-keyboard	15	1	7

24	Internal IC Board	2	0	1
25	Microphone	2	1	4

Table 6. Performance of two types of line on P25

Line type	Algorithm	Evaluation	F_1	F_2	F_3	F_4
Straight-line	SWO	Best value	10	9	80	925
		Avg. value	10.00	9.00	80.00	925.00
		SD	0.00	0.00	0.00	0.00
U-shaped	SWO	Best value	10	9	76	909
		Avg. value	10.00	9.00	77.39	916.28
		SD	0.00	0.00	1.18	6.99

5.2 Comparative study

This section first compares CSO with a variable neighborhood sear method (VNSGA) (Kalayci, Polat, and Gupta, 2016) and an iterated local search method (ILS) (Li, Kucukkoc, and Zhang, 2019) on SDLBP and SUDLBP. Notice that results of VNSGA and ILS are acquired from above mentioned study and only the first two objectives are taken into consideration because of their higher priority. Table 7 reports comparation results of three algorithms. N represents number of tasks.

Table 7. Results of VNSGA, ILS and CSO

Instance	N	CT	VNSGA (SDLBP)		ILS (SDLBP)		CSO (SDLBP)		ILS (SUDLBP)		CSO (SUDLBP)	
			F_1	F_2	F_1	F_2	F_1	F_2	F_1	F_2	F_1	F_2
Mertens	7	7	5	10	5	10	5	10	5	10	5	10
Bowman	8	20	5	149	5	149	5	149	4	13	4	13
Jaeschke	9	7	7	26	7	28	7	28	7	28	7	26
Jackson	11	10	5	6	5	6	5	6	5	4	5	4
Mansoor	11	94	2	5	2	5	2	5	2	5	2	5
Mitchell	21	15	8	31	8	43	8	43	8	29	8	29
Roszieg	25	16	8	5	8	5	8	5	8	3	8	3
Heskiaoff	28	216	5	628	5	630	5	630	5	628	5	628
Buxey	29	30	12	118	12	122	12	118	11	6	11	6
Lutzl	32	2357	7	8.13E+05	7	8.47E+05	7	8.38E+05	7	7.99E+05	7	8.01E+05
Gunther	35	41	14	1519	14	1735	14	1711	12	13	12	13
Kilbridge	45	62	9	6	9	6	9	6	9	6	9	6
Hahn	53	2806	6	1.87E+06	6	1.91E+06	6	1.93E+06	5	6	5	6
Tonge	70	168	22	2152	22	1756	22	2238	22	1672	22	1736
Tonge	70	170	22	3002	22	2660	22	2816	21	204	21	522
Tonge	70	173	22	5196	21	1081	22	4832	21	745	21	972
Tonge	70	179	21	3459	20	312	21	2805	20	262	20	306
Tonge	70	182	20	968	20	912	20	932	20	854	20	880
Wee-Mag	75	46	35	983	34	399	35	617	34	349	34	356
Wee-Mag	75	47	33	148	33	116	33	116	33	106	33	106
Wee-Mag	75	49	32	189	32	163	32	163	32	155	32	149
Wee-Mag	75	50	32	347	32	333	32	339	32	327	32	325
Wee-Mag	75	52	31	455	31	443	31	441	31	431	31	419
Arcus1	83	3985	20	9.34E+05	20	9.22E+05	20	9.19E+05	20	8.14E+05	20	8.07E+05
Arcus1	83	5048	16	1.76E+06	16	1.76E+06	16	1.76E+06	16	1.67E+06	16	1.63E+06
Arcus1	83	5853	14	2.79E+06	14	2.79E+06	14	2.79E+06	13	1.16E+04	13	1.38E+04
Arcus1	83	6842	12	4.26E+06	12	4.25E+06	12	4.26E+06	12	3.43E+06	12	3.41E+06
Arcus1	83	7571	11	5.37E+06	11	5.54E+06	11	5.49E+06	11	5.37E+06	11	5.35E+06
Arcus1	83	8412	10	7.09E+06	10	7.83E+06	10	7.09E+06	10	7.93E+06	10	7.09E+06
Arcus1	83	8898	9	2.14E+06	9	2.15E+06	9	2.14E+06	9	2.13E+06	9	2.13E+06
Arcus1	83	10816	8	1.49E+07	8	3.75E+07	8	1.37E+07	7	1.10E+01	8	1.09E+07
Lutz2	89	15	34	63	34	61	34	63	33	10	33	12
Lutz3	89	150	12	2050	12	2256	12	1892	11	6	11	6
Mukherjee	94	201	23	12057	23	14853	23	11475	21	13	22	977
Mukherjee	94	301	15	10137	15	10137	15	10137	14	6	14	6
Arcus2	111	5755	27	2.58E+06	27	2.40E+06	27	2.38E+06	27	1.06E+06	27	1.04E+06
Arcus2	111	7520	21	3.00E+06	21	2.97E+06	21	2.91E+06	21	2.75E+06	21	2.69E+06
Arcus2	111	8847	18	4.38E+06	18	4.59E+06	18	4.47E+06	18	4.41E+06	18	4.39E+06

Arcus2	111	10027	16	6.33E+06	16	6.39E+06	16	6.37E+06	16	6.42E+06	16	6.31E+06
Arcus2	111	10743	15	7.76E+06	15	7.82E+06	15	7.79E+06	15	7.81E+06	15	7.76E+06
Arcus2	111	11378	14	5.76E+06	14	5.72E+06	14	5.70E+06	14	5.68E+06	14	5.64E+06
Arcus2	111	11570	14	9.86E+06	14	1.02E+07	14	9.86E+06	14	9.63E+06	14	9.49E+06
Arcus2	111	17067	9	1.14E+06	9	1.14E+06	9	1.14E+06	9	1.14E+06	9	1.14E+06
Barthol2	148	85	52	906	51	293	52	863	51	243	51	257
Barthol2	148	89	50	1174	49	425	49	397	48	74	48	92
Barthol2	148	91	49	1179	48	504	48	492	47	67	47	73
Barthol2	148	95	47	1279	46	454	46	426	45	53	45	53

For SDLBP, for minimizing number of workstations (F_1), CSO obtains 3 better and 44 same results compared with VNSGA, and it gets 44 same results compared with ILS. For minimizing total idle times (F_2), CSO has 20 better and 16 same results compared with VNSGA, and it gets 21 better and 15 same results compared with ILS. For SUDLBP, for minimizing number of workstations (F_1), CSO obtains45 same results compared with ILS. For minimizing total idle times (F_2), CSO has 18 better and 17 same results compared with ILS. It is sufficient to conclude that CSO has a great searching ability in solving SDLBP and SUDLBP, also it has superior performance than VNSGA and ILS in many aspects, especially minimizing total idle times. Moreover, U-shaped layout outperforms straight-line disassembly on most of these benchmark instances.

Table 8. Comparation results of F_1 for 10 algorithms

Instance	N	CT	HC	LAHC	SA	TS	GA	ABC	BA	PSO	ILS	CSO
Mertens	7	7	5	5	5	5	5	5	5	5	5	5
Bowman	8	20	4	4	4	4	4	4	4	4	4	4
Jaeschke	9	7	7	7	7	7	7	7	7	7	7	7
Jackson	11	10	5	5	5	5	5	5	5	5	5	5
Mansoor	11	94	2	2	2	2	2	2	2	2	2	2
Mitchell	21	15	8	8	8	8	8	8	8	8	8	8
Roszieg	25	16	8	8	8	8	8	8	8	8	8	8
Heskiaoff	28	216	5	5	5	5	5	5	5	5	5	5
Buxey	29	30	11	11.05	11	11	11	11	11	11	11	11
Lutzl	32	2357	7	7	7	7	7	7	7	7	7	7
Gunther	35	41	12	12	12	12	12	12	12	12	12	12
Kilbridge	45	62	9	9	9	9	9	9	9	9	9	9
Hahn	53	2806	5.7	5.65	5.75	5.65	5.75	5.65	5.85	5.75	5.2	5.25
Tonge	70	168	22	22	22	22.15	22	22	22	22	22	22
Tonge	70	170	21.95	21.95	22	21.95	22	22.15	22	22.15	21.8	21.7
Tonge	70	173	21	21	21.05	21	21	21	21	21	21	21
Tonge	70	179	20	20	20.5	20	20	20	20	20	20	20
Tonge	70	182	20	20	20	20.15	20	20	20	20	20	20
Wee-Mag	75	46	34	34	34.5	34	34	34	34.75	34	34	34
Wee-Mag	75	47	33	33	33	33	33	33	33	33	33	33
Wee-Mag	75	49	32	32	32	32	32	32	32	32	32	32
Wee-Mag	75	50	32	32	32	32	32	32	32	32	32	32
Wee-Mag	75	52	31	31	31	31	31	31	31	31	31	31
Arcus1	83	3985	20	20	20	20	20.15	20	20	20	20	20
Arcus1	83	5048	16	16	16	16	16	16	16.75	16	16	16
Arcus1	83	5853	13	13	13	13.5	13.5	13.85	13.85	13	13	13
Arcus1	83	6842	12	12	12	12	12	12	12	12	12	12
Arcus1	83	7571	11	11	11	11	11	11	11	11	11	11
Arcus1	83	8412	10	10	10	10.4	10	10	10	10.75	10	10
Arcus1	83	8898	9	9	9	9	9	9	9	9	9	9
Arcus1	83	10816	8	8	8	8	8	8	8	8	7.8	8
Lutz2	89	15	33	33	33	33	33	33	33	33.15	33	33
Lutz3	89	150	11	11	11	11.15	11	11	11	11	11	11
Mukherjee	94	201	21.25	21.2	21.4	22	22	22	22	21.75	21.25	22
Mukherjee	94	301	14	14	14.85	14.7	14.9	15	14	14.5	14	14
Arcus2	111	5755	27	27	27	27	27	27	27	27	27	27
Arcus2	111	7520	21	21	21	21	21	21	21	21	21	21
Arcus2	111	8847	18	18	18	18	18	18	18	18	18	18
Arcus2	111	10027	16	16	16	16	16	16	16	16	16	16
Arcus2	111	10743	15	15	15	15	15	15	15	15	15	15

Arcus2	111	11378	14	14	14	14	14	14	14	14	14	14
Arcus2	111	11570	14	14	14	14	14	14	14	14	14	14
Arcus2	111	17067	9	9	9	9	9	9	9	9	9	9
Barthol2	148	85	51	51	51.75	51.05	51.7	51.85	51.15	51.15	51	51
Barthol2	148	89	49	48.9	48.95	49	50.15	49	49.5	49	48.75	48.5
Barthol2	148	91	48	47.8	48.05	48	48.75	48.5	48.75	48	47.6	47.4
Barthol2	148	95	45.9	45.85	46.15	46	45.95	46	46.5	46	45.65	45.5

Table 9. Comparation results of F_2 for 10 algorithms

Instance	N	CT	HC	LAHC	SA	TS	GA	ABC	BA	PSO	ILS	CSO
Mertens	7	7	10	10	10	10	10	10	10	10	10	10
Bowman	8	20	13	13	13	13	13	13	13	13	13	13
Jaeschke	9	7	28	28	28	28	28	28	28	28	28	28
Jackson	11	10	4	4	4	4	4	4	4	4	4	4
Mansoor	11	94	5	5	5	5	5	5	5	5	5	5
Mitchell	21	15	30.7	32	30.1	29.9	30.2	29.7	30	30.1	29.1	29.3
Roszieg	25	16	3.2	3.9	3.0	3.0	3.0	3.0	3.0	3.0	3.0	3.0
Heskiaoff	28	216	634.8	636.4	627.7	633.1	629.9	628.4	628.7	629.0	629.1	628.4
Buxey	29	30	8.4	15.8	6.9	7.3	6.7	8.9	9.2	6.7	6.5	6.4
Lutz l	32	2357	838157	830279	837411	812097	814726	819572	859692	807663	804475	805235
Gunther	35	41	13	13.4	13.9	14.7	13.5	13.4	14.0	13.2	13.1	13.0
Kilbridge	45	62	6.2	8.9	6.2	6.0	6.2	6.3	6.1	6.0	6.0	6.0
Hahn	53	2806	1E+06	1E+06	893754	1E+06	980065	1E+06	1E+06	1E+06	344411	509231
Tonge	70	168	1805.5	1811.3	1833.2	1807.3	1789.1	1787.9	1801.3	1777.1	1783.0	1795.5
Tonge	70	170	2690.9	2651.8	2675.5	2325.0	2680.4	2774.3	2405.9	2217.8	2159.8	1874.5
Tonge	70	173	1088.8	1719.7	867.9	973.2	1204.4	1021.5	937.2	980.3	954.1	1019.3
Tonge	70	179	325.6	518.5	439.0	407.9	390.5	324.2	379.8	300.5	290.8	321.0
Tonge	70	182	934.0	1685.7	979.4	937.5	1235.4	1090.4	873.4	905.9	879.9	901.3
Wee-Mag	75	46	475.4	457.5	442.2	399.5	467.4	492.8	433.0	420.5	426.7	373.5
Wee-Mag	75	47	128.5	118.0	117.4	110.5	105.5	121.2	109.8	123.1	117.3	115.3
Wee-Mag	75	49	159.9	159.5	156.4	159.0	169.3	157.8	163.5	159.4	159.3	156.2
Wee-Mag	75	50	337.8	331.5	337.5	329.9	327.4	356.9	372.7	332.4	330.5	329.4
Wee-Mag	75	52	446.9	444.4	477.5	496.5	442.1	512.3	436.2	463.3	437.8	430.0
Arcus1	83	3985	838896	835347	912370	880923	839762	840125	839227	839972	827898	813055
Arcus1	83	5048	2E+06	2E+06	2E+06	2E+06	2E+06	2E+06	2E+06	2E+06	2E+06	2E+06
Arcus1	83	5853	13515	19389	2E+06	2E+06	1E+06	14909	3E+06	27084	12786	15349
Arcus1	83	6842	4E+06	4E+06	3E+06	4E+06	3E+06	4E+06	4E+06	4E+06	4E+06	4E+06
Arcus1	83	7571	6E+06	6E+06	7E+06	7E+06	6E+06	8E+06	6E+06	6E+06	6E+06	6E+06
Arcus1	83	8412	1E+07	1E+07	1E+07	1E+07	9E+06	1E+07	1E+07	1E+07	1E+07	9E+06
Arcus1	83	8898	2E+06	2E+06	2E+06	2E+06	2E+06	2E+06	2E+06	2E+06	2E+06	2E+06
Arcus1	83	10816	4E+07	4E+07	4E+07	4E+07	3E+07	4E+07	3E+07	4E+07	3E+07	2E+07
Lutz 2	89	15	10.3	16.5	10.9	11.7	12.3	13.2	14.2	11.9	10.1	12.5
Lutz 3	89	150	6.4	10.7	6.9	7.0	92.3	7.7	7.3	6.7	6.6	6.5
Mukherjee	94	201	588.25	475.1	979.2	2107.2	1783.2	973.6	2019.2	1775.3	564.35	1235.0
Mukherjee	94	301	14.4	16.5	17.3	973.2	1987.5	2612.3	785.4	5642.3	9.6	13.9
Arcus2	111	5755	1E+06	2E+06	1E+06	1E+06	1E+06	1E+06	1E+06	1E+06	1E+06	1E+06
Arcus2	111	7520	3E+06	3E+06	3E+06	3E+06	4E+06	3E+06	3E+06	3E+06	3E+06	3E+06
Arcus2	111	8847	5E+06	5E+06	5E+06	6E+06	5E+06	5E+06	5E+06	6E+06	5E+06	5E+06
Arcus2	111	10027	7E+06	7E+06	8E+06	7E+06	8E+06	7E+06	7E+06	7E+06	7E+06	7E+06
Arcus2	111	10743	8E+06	8E+06	9E+06	1E+07	8E+06	9E+06	1E+07	8E+06	8E+06	8E+06
Arcus2	111	11378	6E+06	6E+06	6E+06	6E+06	6E+06	7E+06	6E+06	6E+06	6E+06	6E+06
Arcus2	111	11570	1E+07	1E+07	1E+07	1E+07	1E+07	1E+07	1E+07	1E+07	1E+07	1E+07
Arcus2	111	17067	1E+06	1E+06	1E+06	1E+06	1E+06	1E+06	1E+06	1E+06	1E+06	1E+06
Barthol2	148	85	259.8	258.4	473.1	300.5	512.4	1091.4	298.0	374.5	257.4	265.2
Barthol2	148	89	371.2	346.0	406.5	690.3	394.7	365.8	432.1	386.2	294.65	315.2
Barthol2	148	91	414.0	362.4	397.1	342.0	513.2	468.1	413.4	452.3	281.3	264.2
Barthol2	148	95	419.4	396.95	510.2	437.5	725.6	403.2	597.1	433.0	311.65	307.4

CSO is compared with 9 algorithms and detailed results are listed in Table 8 and Table 9. These algorithms are hill-climbing algorithm (HC) (McGovern and Gupta, 2007a, 2007b), late acceptance hill-climbing algorithm (LAHC) (Yuan, Zhang, and Shao, 2015), simulated annealing algorithm (SA), tabu search algorithm (TS), genetic algorithm (GA), artificial bee colony algorithm (ABC), bee algorithm (BA), particle swarm optimization (PSO), and iterated local search optimization (ILS). Notice that part of data is acquired from related research and SA, TS, GA, ABC, BA, and PSO are re-implemented on a U-shaped disassembly line each for 20 times. Also, results listed are average objective values. Based on the results of comparative study,

CSO has a strong global searching ability and its mechanism help it avoid trapping into local optimal field. CSO has a superior performance in solving large-size instances especially Barthol 2 instance.

6. Conclusion

Disassembly line balancing problem (DLBP) has become an active research area since the raising attention of globally environment protection. As a critical step of product recovery, disassembly process needs more attentions in the future. DLBP is not the verse process of assembly line balancing problem (ALBP), and it is much more complex. This study has for the first time proposed a novel meta-heuristic algorithm (CSO) on a U-shaped disassembly line with the consideration of sequence-dependent situation. An exhaustive mixed-integer non-linear programming model (MINLP) is proposed which can solve different precedence relationships. This study presented comparation results of 10 algorithms, CSO outperforms other compared algorithms in many aspects, especially minimizing total idle times. Novel algorithms are welcomed, and combined methods and approaches are attractive in future research.

U-shaped layout has higher flexibility, and it is not considered as many as straight-line disassembly line. Therefore, U-shaped line together with parallel and two-sided line are interesting areas to explore.Due to the uncertainty of DLBP, non-deterministic and fuzzy DLBP will be suggested to applied. Also, special types of DLBP, like sequence dependent and partial DLBP are novel area to study.

References

[1]. Agrawal, S. and Tiwari, M.K., 2008. A collaborative ant colony algorithm to stochastic mixed-model U-shaped disassembly line balancing and sequencing problem. *International Journal of Production Research*, *46*(6), pp.1405-1429.

[2]. Akpinar, M.E., Ilgin, M.A. and Aktaş, H., 2021. Disassembly Line Balancing by Using Simulation Optimization. *Alphanumeric Journal*, *9*(1), pp.63-84.

[3]. Avikal, S., Jain, R. and Mishra, P., 2013. A heuristic for U-shaped disassembly line balancing problems. *MIT International Journal of Mechanical Engineering*, *3*(1), pp.51-56.

[4]. Avikal, S. and Mishra, P.K., 2012. A new U-shaped heuristic for disassembly line balancing problems. *PRATIBHA: International Journal of Science, Spirituality, Business and Technology*, *1*(1), pp.21-27.

[5]. Chen, J.C., Chen, Y.Y., Chen, T.L. and Yang, Y.C., 2021. An adaptive genetic algorithm-based and AND/OR graph approach for the disassembly line balancing problem. *Engineering Optimization*, pp.1-17.

[6]. Chu, S.C., Tsai, P.W. and Pan, J.S., 2006, August. Cat swarm optimization. In *Pacific Rim international conference on artificial intelligence* (pp. 854-858). Springer, Berlin, Heidelberg.

[7]. Edis, E.B., Edis, R.S. and Ilgin, M.A., 2022. Mixed integer programming approaches to partial disassembly line balancing and sequencing problem. *Computers & Operations Research*, *138*, p.105559.

[8]. Gao, Y., Lou, S., Zheng, H. and Tan, J., 2021. A data-driven method of selective disassembly planning at end-of-life under uncertainty. *Journal of Intelligent Manufacturing*, pp.1-21.

[9]. Gungor, A., & Gupta, S. M., 1999a. Disassembly line balancing. *Proceedings of the 1999 Annual Meeting of the Northeast Decision Sciences Institute, Newport*, Rhode Island, March 24-26, pp.193-195.

[10]. Gungor, A. and Gupta, S.M., 1999b. Issues in environmentally conscious manufacturing and product recovery: a survey. *Computers & Industrial Engineering*, 36(4), pp.811-853.

[11]. Kalayci, C.B. and Gupta, S.M., 2013a. Artificial bee colony algorithm for solving sequence-dependent disassembly line balancing problem. *Expert Systems with Applications*, *40*(18), pp.7231-7241.

[12]. Kalayci, C.B. and Gupta, S.M., 2013b. A particle swarm optimization algorithm with neighborhood-based mutation for sequence-dependent disassembly line balancing problem. *The International Journal of Advanced Manufacturing Technology*, *69*(1), pp.197-209.

[13]. Kalayci, C.B. and Gupta, S.M., 2013c. Simulated annealing algorithm for solving sequence-dependent disassembly line balancing problem. *IFAC Proceedings Volumes*, *46*(9), pp.93-98.

[14]. Kalayci, C.B. and Gupta, S.M., 2014. A tabu search algorithm for balancing a sequence-dependent disassembly line. *Production Planning & Control*, *25*(2), pp.149-160.

[15]. Kalayci, C.B., Polat, O. and Gupta, S.M., 2016. A hybrid genetic algorithm for sequence-dependent disassembly line balancing problem. *Annals of Operations Research*, *242*(2), pp.321-354.

[16]. Li, Z. and Janardhanan, M.N., 2021. Modelling and solving profit-oriented U-shaped partial disassembly line balancing problem. Expert Systems with Applications, p.115431.

[17]. Li, Z., Kucukkoc, I., Tang, Q. and Zhang, Z., 2021. Models and two-phase bee algorithms for multi-objective U-shaped disassembly line balancing problem. *Optimization and Engineering*, pp.1-32.

[18]. Li, Z., Kucukkoc, I. and Zhang, Z., 2019. Iterated local search method and mathematical model for sequence-dependent U-shaped disassembly line balancing problem. *Computers & Industrial Engineering*, *137*, p.106056.
[19]. McGovern, S.M. and Gupta, S.M., 2007a. A balancing method and genetic algorithm for disassembly line balancing. *European journal of operational research*, *179*(3), pp.692-708.
[20]. McGovern, S.M. and Gupta, S.M., 2007b. Combinatorial optimization analysis of the unary NP-complete disassembly line balancing problem. *International Journal of Production Research*, *45*(18-19), pp.4485-4511.
[21]. Xu, S., Guo, X., Liu, S., Qi, L., Qin, S., Zhao, Z. and Tang, Y., 2021, October. Multi-objective Optimizer with Collaborative Resource Allocation Strategy for U-shaped Stochastic Disassembly Line Balancing Problem. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 2316-2321). IEEE.
[22]. Özceylan, E., Kalayci, C.B., Güngör, A. and Gupta, S.M., 2019. Disassembly line balancing problem: a review of the state of the art and future directions. *International Journal of Production Research*, *57*(15-16), pp.4805-4827.
[23]. Wang, K., Li, X., Gao, L., Li, P. and Gupta, S.M., 2021. A genetic simulated annealing algorithm for parallel partial disassembly line balancing problem. *Applied Soft Computing*, *107*, p.107404.
[24]. Wang, S., Guo, X. and Liu, J., 2019. An efficient hybrid artificial bee colony algorithm for disassembly line balancing problem with sequence-dependent part removal times. *Engineering Optimization*, *51*(11), pp.1920-1937.
[25]. Wang, W., Guo, X., Liu, S., Qin, S.J., Qi, L., Zhao, Z. and Tang, Y., 2021, October. Multi-objective Discrete Chemical Reaction Optimization Algorithm for Multiple-product Partial U-shaped Disassembly Line Balancing Problem. In *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 2322-2327). IEEE.
[26]. Yao, P. and Gupta, S. M., 2021a. Cat Swarm Optimization Algorithm for Solving Multi-Objective U-Shaped Disassembly Line Balancing Problem. *Proceedings of the International Conference on Remanufacturing*, March 24-25, pp. 222-230.
[27]. Yao, P. and Gupta, S. M., 2021b. Small World Optimization Algorithm for Solving Multi-Objective U-Shaped Disassembly Line Balancing Problem, *Proceedings of the 2021 Annual Meeting of the Northeast Decision Sciences Institute*, Virtual, March 26-27, 659-668.
[28]. Yao, P. and Gupta, S. M., 2021c. Ant Colony Optimization Algorithm for Solving U-Shaped Disassembly Line Balancing Problem with Multiple Objectives, *Proceedings of the 4th International Conference on Innovative Studies of Contemporary Sciences*, Tokyo, Japan, July 29-31, pp. 21-26.
[29]. Yao, P. and Gupta, S. M., 2021d. Invasive Weed Optimization Algorithm for Solving Multi-Objective U-Shaped Disassembly Line Balancing Problem", *Proceedings of the 12th International Congress on Mathematics, Engineering and Natural Sciences*, Paris, France, July 9-11, pp. 286-292.
[30]. Yao, P. and Gupta, S. M., 2021e. Teaching-Learning-Based Optimization Algorithm for Solving Multi-Objective U-Shaped Disassembly Line Balancing Problem, *Proceedings of the 5th International New York Conference on Evolving Trends in Interdisciplinary Research and Practices*, Manhattan, New York City, October 3-5, pp. 21-28.
[31]. Yao, P. and Gupta, S. M., 2021f. Fish School Search Algorithm for Solving Multi-Objective U-Shaped Disassembly Line Balancing Problem, *Proceedings of the Latin American International Conference on Natural and Applied Sciences*, Villahermosa, Mexico, November 5-6, pp. 44-52.
[32]. Yin, T., Zhang, Z., Zhang, Y., Wu, T. and Liang, W., 2022. Mixed-integer programming model and hybrid driving algorithm for multi-product partial disassembly line balancing problem with multi-robot workstations. *Robotics and Computer-Integrated Manufacturing*, *73*, p.102251.
[33]. Yuan, B., Zhang, C. and Shao, X., 2015. A late acceptance hill-climbing algorithm for balancing two-sided assembly lines with multiple constraints. *Journal of Intelligent Manufacturing*, *26*(1), pp.159-168.
[34]. Zhang, Z., Wang, K., Zhu, L. and Wang, Y., 2017. A Pareto improved artificial fish swarm algorithm for solving a multi-objective fuzzy disassembly line balancing problem. *Expert Systems with Applications*, *86*, pp.165-176.
[35]. Zhu, X., Zhang, Z., Zhu, X. and Hu, J., 2014. An ant colony optimization algorithm for multi-objective disassembly line balancing problem. *China Mechanical Engineering*, *25*(8), p.1075.