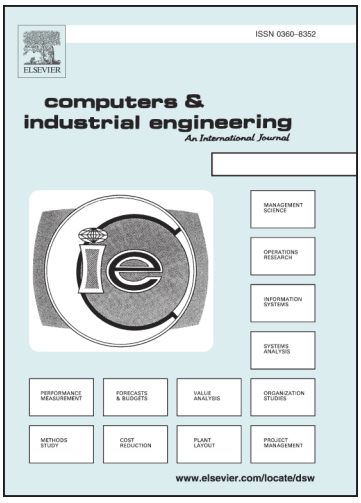
Journal Pre-proofs



Iterated local search method and mathematical model for sequence-dependent U-shaped disassembly line balancing problem

Zixiang Li, Ibrahim Kucukkoc, Zikai Zhang

PII: S0360-8352(19)30515-7
DOI: https://doi.org/10.1016/j.cie.2019.106056
Reference: CAIE 106056

To appear in: *Computers & Industrial Engineering*

Received Date: 15 November 2018
Revised Date: 6 August 2019
Accepted Date: 8 September 2019

Please cite this article as: Li, Z., Kucukkoc, I., Zhang, Z., Iterated local search method and mathematical model for sequence-dependent U-shaped disassembly line balancing problem, *Computers & Industrial Engineering* (2019), doi: https://doi.org/10.1016/j.cie.2019.106056

Iterated local search method and mathematical model for sequence-dependent U-shaped disassembly line balancing problem

Zixiang Li[1,2], Ibrahim Kucukkoc[3*], Zikai Zhang [1,2]

[1]*Key Laboratory of Metallurgical Equipment and Control Technology, Wuhan University of Science and Technology, Wuhan, Hubei, China*

[2]*Hubei Key Laboratory of Mechanical Transmission and Manufacturing Engineering, Wuhan University of Science and Technology, Wuhan, Hubei, China*

[3]*Industrial Engineering Department, Balikesir University, Cagis Campus, Balikesir 10145, Turkey*

Email: zixiangliwust@gmail.com (Z. Li); ikucukkoc@balikesir.edu.tr (I. Kucukkoc); zhangzikai0703@gmail.com (Z. Zhang);

**Corresponding author. Tel: +902666121194 (Ext. 6407)*

Abstract

Disassembly lines play an important role in disassembling end-of-life products and retrieving the valuable components. Sequence-dependent time increments are inevitable due to the interactions between two tasks in some occasions. This research provides the first attempt to study the sequence-dependent U-shaped disassembly line balancing problem (SUDLBP) with multiple objectives. A mixed-integer programming model is developed to solve the small-size instances optimally, where new expressions are developed to formulate the AND/OR precedence relations. Due to the NP-hard nature of this problem, a simple and effective iterated local search algorithm is developed for the first time. This algorithm utilizes modified NEH heuristic to achieve a high-quality initial solution. A new local search technic with referenced permutation and two neighbor structures is also embedded into it to obtain the local optimum solution. Computational tests demonstrate that the U-shaped lines outperform traditional straight lines in terms of the objective values. The comparative study on two sets of instances demonstrates that the proposed method outperforms the CPLEX solver in search speed and achieves a competing performance in comparison with other eight re-implemented algorithms.

Keywords: Disassembly line balancing; Sequence-dependent U-shaped disassembly line; Integer programming; Iterated local search algorithm; Metaheuristics

Iterated local search method and mathematical model for sequence-dependent U-shaped disassembly line balancing problem

Abstract

Disassembly lines play an important role in disassembling end-of-life products and retrieving the valuable components. Sequence-dependent time increments are inevitable due to the interactions between two tasks in some occasions. This research provides the first attempt to study the sequence-dependent U-shaped disassembly line balancing problem (SUDLBP) with multiple objectives. A mixed-integer programming model is developed to solve the small-size instances optimally, where new expressions are developed to formulate the AND/OR precedence relations. Due to the NP-hard nature of this problem, a simple and effective iterated local search algorithm is developed for the first time. This algorithm utilizes modified NEH heuristic to achieve a high-quality initial solution. A new local search technic with referenced permutation and two neighbor structures is also embedded into it to obtain the local optimum solution. Computational tests demonstrate that the U-shaped lines outperform traditional straight lines in terms of the objective values. The comparative study on two sets of instances demonstrates that the proposed method outperforms the CPLEX solver in search speed and achieves a competing performance in comparison with other eight re-implemented algorithms.

Keywords: Disassembly line balancing; Sequence-dependent U-shaped disassembly line; Integer programming; Iterated local search algorithm; Metaheuristics

1. Introduction

Recently, there is an increasing number of obsolete products especially with short product life-cycles decreasing constantly. Improper dealing with the end-of-life (EOL) products yields not only the large waste but also environmental problems. Product recovery has become even more popular and urgent. It is usually realized through recycling, refurbishing or remanufacturing the valuable components, where disassembly is the most critical and time-consuming step. Disassembly process separates the EOL products into components for reuse or remanufacturing. Disassembly lines have been widely employed in industry thanks to their higher productivity, easy training of the workers and suitability for automation (Güngör and Gupta 1999, Gungor and Gupta 2001). The components are separated on a set of paced stations linked to each other via a material transportation system. If the disassembly line is not well designed, there will be a large amount of wasted time and line imbalance. A highly efficient disassembly line may not be a global solution to recycle all EOL products. However, it may play an enormously important role in dealing with the short product cycles especially in the electronics sector in today's modern industrial environment. Hence, the topic of balancing the disassembly lines, known as disassembly line balancing problem (DLBP), attracts academics and practitioners.

In the DLBPs, disassembly operations cannot be regarded as the reverse operations of the assembly operations due to the unique characteristics. There are much complex precedence relations in DLBP, including AND precedence, OR precedence and complex AND/OR precedence (Güngör and Gupta 2002). There is a high degree of uncertainty in the quality, reliability and conditions of the EOL products in the disassembly process. In addition, there are some hazardous tasks needing special handling and

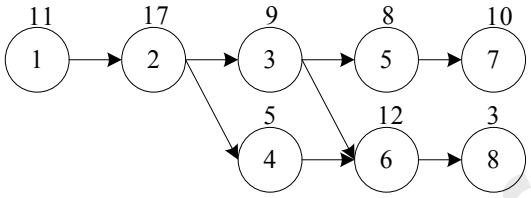
they are preferred to be removed at early stations if possible.

Since the pioneering works published by (Güngör and Gupta 1999) and (Gungor and Gupta 2001), the DLBP has drawn an increasing attention. Many heuristics, metaheuristics and exact methods were developed for this problem, which was proven to be NP-hard by (McGovern and Gupta 2007) and (McGovern and Gupta 2007). Heuristic methods include the heuristic presented by (Güngör and Gupta 2002), 2-opt hybrid algorithm introduced by (McGovern and Gupta 2003) and later employed by (Ren, Zhang et al. 2018), and a beam search heuristic by (Mete, Çil et al. 2016). The beam search heuristic produced a promising performance in solving large-size problems. Exact methods mainly have the mixed integer linear programming model (MILP) utilizing CPLEX solver (Altekin, Kandiller et al. 2008, Koc, Sabuncuoglu et al. 2009, Altekin and Akkan 2012, Paksoy, Güngör et al. 2013, Mete, Çil et al. 2018), which is capable of obtaining the optimal solution for small-size instances. Nevertheless, one main drawback of these MILP models is that it might cost tremendous time to solve large-size instances. Sometimes it is not even possible to get satisfying results within an acceptable time. Hence, metaheuristics were applied to obtain near-optimal solutions and solve the multi-objective DLBPs. These include genetic algorithm (GA) (McGovern and Gupta 2007, Kalayci, Polat et al. 2016, Ren, Zhang et al. 2018), ant colony optimization (McGovern and Gupta 2006, Agrawal and Tiwari 2008, Ding, Feng et al. 2010, Kalayci and Gupta 2013), artificial bee colony (Kalayci and Gupta 2013, Kalayci, Hancilar et al. 2015, Liu and Wang 2017), tabu search (TS) (Kalayci and Gupta 2014) and particle swarm optimization (PSO) (Kalayci and Gupta 2013, Xiao, Wang et al. 2017). Also, variable neighborhood search (Ren, Zhang et al. 2018), gravitational search algorithm (Ren, Yu et al. 2017), artificial fish swarm algorithm (Zhang, Wang et al. 2017), bees algorithm (Liu, Zhou et al. 2018), and firefly algorithm (Zhu, Zhang et al. 2018) were applied to DLBPs. For in-depth reviews, please refer to Tsao (2015), Hoseini et al. (2019), Gharaei, Karimi et al. (2019), Gharaei, Hoseini et al. (2019), Rabbani et al. (2018, 2019), Kazemi et al. (2018) and Gharaei, Karimi et al. (2019).

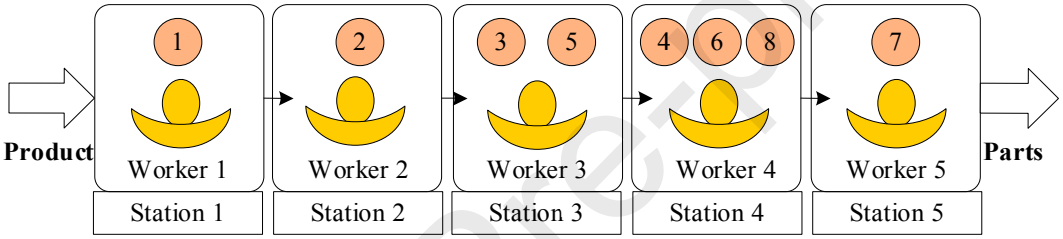
In real-world applications, there might be interactions between tasks in some occasions. The removal time of a part might be influenced by the removal of another part, resulting in the sequence-dependent disassembly line balancing problem (SDLBP). This topic has an importance to prevent any delays on the line and also increase the efficiency. However, sequence dependency has been studied only for the straight disassembly lines (Özceylan, Kalayci et al. 2018). For example, (Kalayci and Gupta 2013) introduced this problem and described it with a mathematical formulation, and presented an artificial bee colony (ABC) algorithm. However, the presented model to describe this problem cannot be solved utilizing an exact technique. Later, they presented a PSO algorithm (Kalayci and Gupta 2013) and a TS algorithm (Kalayci and Gupta 2014). More recently, (Kalayci, Polat et al. 2016) re-formulated this model and presented a hybrid GA. Still, this new model cannot be solved utilizing an exact technique. (Liu and Wang 2017) proposed an improved ABC algorithm, which was stated to outperform all the aforementioned algorithms in the computational study.

U-shaped lines differ from the straight lines based on their U-shaped layout, which brings many advantages based on the increased possibility of allocating tasks to workstations in different combinations. In a U-shaped line, a worker in the first workstation may perform tasks from both the beginning and end of the precedence relationship diagram, which also increases the problem complexity. Figure 1 illustrates an example, with eight parts and a cycle time of 20 units, to highlight the features of U-shaped disassembly lines. The precedence diagram of the problem is provided in Fig. 1(a). Fig. 1(b) and Fig. 1(c) present the optimal task assignments on a straight disassembly line and on a U-shaped disassembly line, respectively, in terms of the number of workers. On a straight disassembly line, the

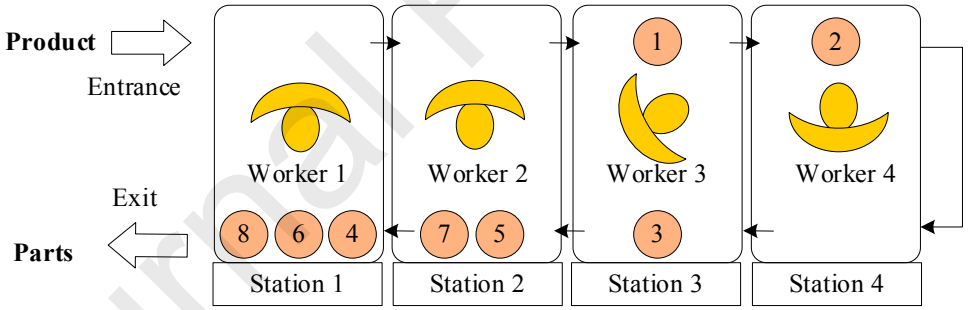
product is disassembled from the first station to the last station until all the parts are removed. On a U-shaped disassembly line, one workstation is divided into two sub-stations: the sub-station on the entrance side and the sub-station on the exit side. As presented in Fig. 1(c), the product is disassembled in the sequence of the sub-stations on the entrance side of station 3 and station 4; and the sub-stations on the exist side of station 3, station 2, and station 1. Clearly, the product is disassembled from the entrance side to the exit side until all the parts are removed. Worker 3 first completes task 1 on the entrance side, later operates task 3 on the exit side and afterwards comes back to operate task 1 on the entrance side again. A clear advantage of the U-shaped layout is that fewer workers may be needed due to higher flexibility. In this example, the U-shaped disassembly line needs only four workers whereas the straight disassembly line needs five workers.



(a) Precedence diagram of an 8-part product



(b) Optimal task assignment on a straight disassembly line



(c) Optimal task assignment on a U-shaped disassembly line

Figure 1 An illustrated example for the comparison of straight and U-shaped lines

The studies on U-shaped disassembly lines are extremely limited and none of them deals with the sequence dependency. There are mainly three studies related to the U-shaped disassembly line balancing problem (UDLBP): one study addressed the stochastic mixed-model UDLBP with an ant colony algorithm (Agrawal and Tiwari 2008) and two studies addressed the UDLBP with heuristics (without sequence dependency) (Avikal and Mishra 2012, Avikal, Jain et al. 2013). As observed from this survey, there is a gap in the literature for sequence-dependent U-shaped disassembly line balancing problem (SUDLBP).

For all aforementioned reasons above, this work provides three main contributions to the literature:

1. A mixed-integer programming model is developed to formulate the SUDLBP and solve it optimally. This model attributes the modelling techniques in SDLBP and SUDLBP in two aspects.

Firstly, the proposed model is a mixed-integer programming model that is capable of optimizing the number of stations utilizing CPLEX solver or optimizing a non-linear objective utilizing the mixed-integer non-linear programming (MINLP) solver. Nevertheless, none of the published models on SDLBP (Kalayci and Gupta 2013, Kalayci and Gupta 2013, Kalayci and Gupta 2014, Kalayci, Polat et al. 2016, Wang, Guo et al. 2019), to our best knowledge, can solve the small-size instances optimally utilizing an exact technique. Technically, these published models could be regarded as unsuitable and might lead to possible confusion or misunderstanding. Secondly, this study presents new expressions to deal with the AND/OR precedence relations by diving one station into two sub-stations. Recall that, none of the published studies on U-shaped disassembly lines (Agrawal and Tiwari 2008, Avikal and Mishra 2012, Avikal, Jain et al. 2013) has formulated the precedence relations via mixed-integer programming.

2. This study presents a simple and effective iterated local search (ILS) algorithm to tackle the SUDLBP, which is classified as NP-hard in nature. The reason for selecting ILS is its simplicity in implementation and superior performance in kinds of different combinational optimization problems (Lourenço, Martin et al. 2003, Stützle 2006, Pan and Ruiz 2012). To the best of the authors' knowledge, this is the first attempt to utilize ILS to solve SUDLBP or even DLBP. To solve the SUDLBP effectively, the proposed ILS proposes several modifications. Firstly, a new decoding procedure is developed to obtain a feasible solution, where the assigned task set is divided into two sub-sets on the entrance side and the exit side. Secondly, the NEH heuristic is extended and modified to obtain a high-quality initial solution, which produces superior performance than the well-known simple heuristics. Thirdly, the proposed ILS utilizes a new local search procedure with referenced permutation and two neighbor structures to obtain a local optimum solution. This new local search procedure outperforms published ones (Ruiz and Stützle 2007, Pan and Ruiz 2014).
3. A set of eight metaheuristics are extended to solve the SUDLBP and a comprehensive study is carried out on two sets of instances to evaluate the U-shaped disassembly line and proposed ILS method. Notice that, this is also the first time to apply these metaheuristics to SUDLBP. Computational tests show that the U-shaped line obtains a better line balance than the traditional straight line. The comparative study demonstrates that the improvements enhance the search capacity of ILS by a significant margin. The proposed ILS algorithm outperforms the CPLEX solver in search speed and achieves competing performance in comparison with these re-implemented algorithms.

The remainder of this study is structured as follows. Section 2 presents the problem statement along with the detailed model formulation. Subsequently, the description of the proposed algorithm is provided in detail in Section 3. Section 4 presents the case studies and the comparative study to test the performance of the proposed algorithm. Finally, Section 5 concludes this study and gives several future research venues.

2. Problem statement

This section introduces the considered problem and later presents the detailed mathematical formulation.

2.1 Problem definition

DLBP involves assigning a set of disassembly tasks to several stations with one or several optimization criteria, including minimizing the station number, optimizing the line balance and removing hazardous tasks or tasks with larger demand at earlier stations. Figure 2 illustrates an 8-part PC example by (Kalayci

and Gupta 2014), and each part needs a positive removing time presented in Table 1. There are two main constraints needed to be satisfied. Cycle time constraint requires that the total operation time of tasks on each station does not exceed the cycle time given. Precedence constraint indicates that all the AND predecessors or at least one OR predecessor must be allocated before this task (this example only has AND predecessors) (Güngör and Gupta 2002).

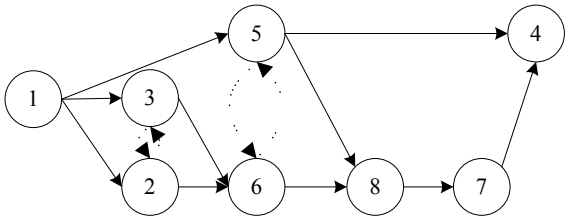


Figure 2 Precedence diagram of 8-part PC instance (sequence dependency in dashed line arrows)

Table 1 Database for the 8-part PC instance by (Kalayci and Gupta 2014)

Task	Part name	Part removal time	Hazardous	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

In real-world applications, an interaction between tasks might happen and the operation times may be influenced by the remained parts/tasks (Kalayci and Gupta 2013, Kalayci and Gupta 2013, Kalayci and Gupta 2014). This situation might also happen between two tasks with no precedence relationship in between, when one component hinders the other component. So that additional removing time is consumed. Let us assume that task i and task j have no precedence relation in between and task i is operated before task j. The disassembly time of task i is affected by task j and sd_{ji} time units is added to compute the operation time of task i. Here, the sequence dependencies of the 8-part PC instance are provided as follows: $sd_{2,3} = 2, sd_{3,2} = 4$; $sd_{5,6} = 1, sd_{6,5} = 3$. Suppose that the task operation sequence is 1, 2, 3, 6, 5, 8, 7, 4. The real removing time of part 2 is $t_2 + sd_{3,2} = 10 + 4$, and the real removing time of part 6 is $t_6 + sd_{5,6} = 16 + 1$ due to the sequence dependencies.

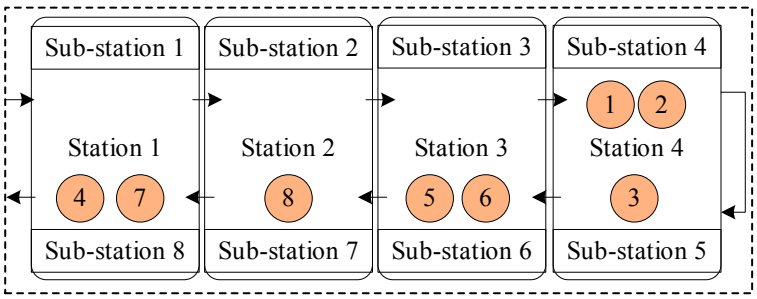


Figure 3 Task assignment on a U-shaped disassembly line

Compared with the straight lines, U-shaped lines have higher flexibility and line efficiency (Agrawal and Tiwari 2008, Avikal and Mishra 2012, Avikal, Jain et al. 2013), where a task might be allocated to either entrance side or exit side while satisfying cycle time constraint and precedence constraint. Figure 3 illustrates one task assignment on the U-shaped disassembly line with sequence dependencies. Due to the sequence dependencies, the total operation time of station 4 is $t_1 + t_2 + sd_{3,2} + t_3 = 14 + 10 + 4 + 12$ and the total operation time of station 3 is $t_6 + sd_{5,6} + t_5 = 16 + 1 + 23$.

2.2 Model formulation

This section presents the mathematical model for the SUDLBP with AND/OR precedence. Notations utilized in the formulation are introduced as follows.

Indices and parameters:

i, j, h	Task/part index, $i,j \in \{1,2,\cdots,N\}$, where N is the number of tasks/parts.
m,n	Sub-station index, $m,n \in \{1,2,\cdots,2M\}$, where M is the maximum number of stations allowed to be opened.
CT	Given cycle time.
t_i	Operation time of performing task i.
h_i	1, if part i is hazardous; 0, otherwise.
d_i	Demand; requested quantity of part i.
$\mathrm{ANDP}(i)$	Set of AND predecessor of task i.
$\mathrm{ORP}(i)$	Set of OR predecessor of task i.
ORPT	Set of tasks which have OR predecessors.
sd_{ji}	Sequence dependent time increment influence of task j on task i.
SD	Set of interacting tasks, $SD = \{(i,j)\}$.
SD_i	Set of tasks interacting with task i.

Decision variables:

T_m	The completion time of station m.
x_{im}	1, if task i is allocated to substation m; 0, otherwise.
y_{imj}	1, if task i is allocated to substation m and is executed before task j; 0, otherwise.
w_{ij}	1, if task i is executed before task j; 0, otherwise.
s_i	Sequence of task i in the solution.
z_m	1, if station m is opened; 0, otherwise.

On the basis of (Kalayci, Polat et al. 2016), the model formulation is provided as follows to optimize four objectives utilizing Equations (1)-(19). Here, the sub-stations are numbered and encoded with $[1,2,\cdots,2M]$ in dealing with the precedence relations (see Figure 3). Specifically, the sub-stations on the entrance side are coded with $[1,2,\cdots,M]$ in the increasing order of the station indexes; the sub-stations on the exit side are coded with $[2 \times M,\ 2 \times M - 1,\ ...,M + 2,M + 1]$ in the decreasing order of the station indexes.

$$\text{Min } f_1 = \sum_{m=1}^{M} z_m \tag{1}$$

$$\text{Min } f_2 = \sum_{m=1}^{M} z_m \cdot (CT - T_m)^2 \quad (2)$$

$$\text{Min } f_3 = \sum_{i=1}^{N} (s_i \cdot h_i) \quad (3)$$

$$\text{Min } f_4 = \sum_{i=1}^{N} (s_i \cdot d_i) \quad (4)$$

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

$$T_m \le z_m.CT \ \ \forall m \quad (6)$$

$$\sum_{i=1}^{N} t_i \cdot (x_{im} + x_{i,2M+1-m}) + \sum_{i=1}^{N}\sum_{j=1}^{N} sd_{ji} \cdot (y_{imj} + y_{i,2M+1-m,j}) = T_m \ \ \forall i; \forall (j,i) \in SD \quad (7)$$

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

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

$$x_{im} + w_{ij} \le 1 + y_{imj} \ \ \forall i; \forall\, j; \forall m; \forall (j,i) \in SD \quad (10)$$

$$x_{im} \ge y_{imj} \ \ \forall i; \forall\, j; \forall m; \forall (j,i) \in SD \quad (11)$$

$$w_{ij} \ge y_{imj} \ \ \forall i; \forall\, j; \forall m; \forall (j,i) \in SD \quad (12)$$

$$w_{ii} = 0 \ \ \forall i \quad (13)$$

$$w_{ij} + w_{ji} = 1 \ \forall i,j \text{ and } i < j \quad (14)$$

$$w_{ji} = 1 \ \ \forall j \in \text{ANDP}(i) \quad (15)$$

$$\sum_{j \in \text{ORP}(i)} w_{ji} \ge 1 \ \ \forall i \in \text{ORPT} \quad (16)$$

$$x_{im} + x_{jn} - 1 \le w_{ij} \ \ \forall i,j,m,n, m < n \quad (17)$$

$$w_{ih} + w_{hj} - 1 \le w_{ij} \ \ \forall i,j,h, i \ne h, h \ne j \text{ and } i \ne j \quad (18)$$

$$s_i = N - \sum_{j=1}^{N} (w_{ij}) \quad (19)$$

In this model, Equation (1) minimizes the number of opened stations. Equation (2) optimizes the line balance or the idle time distribution, and this is a non-linear objective. Equation (3) indicates that the hazardous parts should be removed at first, and Equation (4) ensures that the parts with larger demand are removed at earlier workstations. Constraint (5) indicates that a task must be allocated to the entrance side or exit side of one station. Constraint (6) and Constraint (7) handle the cycle time constraint, indicating that the completion time (including task operation times and sequence dependent times) should be less than or equal to the cycle time. Constraint (8) and Constraint (9) deal with the precedence constraint. Constraint (8) ensures that the AND predecessors of a task should be allocated to the former

or the same station. Constraint (9) ensures that at least one of the OR predecessors should be allocated to the former or the same station. Constraints (10-12) connects the x_{im}, w_{ij} and y_{imj}, indicating that the y_{imj} takes the value of 1 when task i is allocated to sub-station m and task i is executed before task j. Constraints (13-18) tackle the calculation of the w_{ij}. Specifically, constraint (14) indicates that task i should be operated before task j or task j should be operated before task i. Constraint (15) indicates that w_{ji} is equal to 1 when task j is the AND predecessor of task i. Constraint (16) ensures that the sum of $\sum_{j \in \mathrm{ORP}(i)} w_{ji}$ is larger than or equal to 1 when task j is one of the OR predecessors of task i. Constraint (17) indicates that w_{ij} is equal to 1 when task i is allocated to former station than task j. Constraint (18) means that task i is executed before task j when task i is executed before task h and task h is executed before task j. Constraint (19) calculates the sequence of task i in the solution.

Recall that the proposed model differentiates from the published models on SDLBP in two aspects. Firstly, the proposed model is a mixed-integer programming model and this model is capable of solving the small-size instances optimally utilizing the CPLEX solver for the linear objective or MINLP solver for the non-linear objective. On the contrary, the published models for SDLBP (Kalayci and Gupta 2013, Kalayci and Gupta 2013, Kalayci and Gupta 2014, Kalayci, Polat et al. 2016, Wang, Guo et al. 2019) only help describe the problem and none of the published models is capable of solving the small-size instances optimally. Technically, all these published models could be regarded as inapplicable. Moreover, the published models for SDLBP only consider the AND precedence relation, whereas the proposed model deals with AND precedence relation with Constraint (8) and the OR precedence relation with Constraint (9) properly. The model is also capable of solving the instances with only AND precedence relations via disabling Constraint (9).

Secondly, this is the first mathematical model for the SUDLBP and it utilizes Constraint (5), Constraint (8) and Constraint (9) to handle the AND/OR precedence relations. Nevertheless, none of the published studies on U-shaped disassembly line (Agrawal and Tiwari 2008, Avikal and Mishra 2012, Avikal, Jain et al. 2013) has formulated the precedence relations utilizing the mixed-integer programming. For the models on U-shaped assembly lines, they usually utilize Equations (20)-(22) to deal with the AND precedence relations (Urban 1998, Urban and Chiang 2006) (see the cited papers for a detailed description). Here, p is the station index and A_{ip} and B_{jp} are the binary variables to describe the task assignment on the entrance side and exit side; respectively. Here, $A_{ip} = 1$ when task i is allocated to entrance side of station p; and $A_{ip} = 0$, otherwise. $B_{ip} = 1$ when task i is allocated to exit side of station p; $B_{ip} = 0$, otherwise.

$$\sum_{p}^{M} (A_{ip} + B_{jp}) = 1 \quad \forall i \tag{20}$$

$$\sum_{p}^{M} (M - p - 1) \cdot (A_{ip} - A_{jp}) = 1 \quad \forall i; \forall i \in \mathrm{ANDP}(j) \tag{21}$$

$$\sum_{p}^{M} (M - p - 1) \cdot (B_{jp} - B_{ip}) = 1 \quad \forall i; \forall i \in \mathrm{ANDP}(j) \tag{22}$$

Clearly, this published model divides one station into two sub-stations and utilizes A_{ip} and B_{jp} to describe the task assignment on the entrance side and exit side. Although the basic idea is similar, the proposed model encodes the sub-stations with $[1,2,\cdots,2M]$ (see Figure 3). The precedence relation is satisfied when all the AND predecessors and at least one of the OR predecessors of one task are assigned

to the former (with larger index) or the same sub-station. Notice that the proposed model has the same number of variables as the published one ($\mathrm{N} \cdot \mathrm{M} + \mathrm{N} \cdot \mathrm{M} = \mathrm{N} \cdot (2 \cdot M)$). Another difference between the two approaches is that the proposed model is capable of dealing with the AND/OR precedence relations, whereas the published one again cannot handle the OR precedence relation.

Following (McGovern and Gupta 2006), (Kalayci and Gupta 2013) and many others, this research utilizes the hierarchy method in (McGovern and Gupta 2006) to handle these objectives: f_1 has the highest priority, f_2 has the second highest priority and it takes effect when f_1 cannot be further improved. The third objective f_3 has the third highest priority and it takes effect when f_1 and f_2 cannot be further improved. Finally, f_4 has the lowest priority and it takes effect when f_1, f_2 and f_3 cannot be further improved. Among the objectives and constraints, only f_2 in Equation (2) is non-linear and it is applicable to optimize other objectives utilizing CPLEX solver. Recall that, when utilizing the MINLP solver to solve several objectives simultaneously, this research utilizes the weighting method and the objectives with higher priorities are provided with much larger weights.

3. Proposed iterated local search algorithm

As the considered problem is NP-hard in the strong sense, the developed model in Section 2.2 fails to solve the large-size instances efficiently due to tremendous computation time. Hence, this research also develops an ILS algorithm to tackle the SUDLBP within acceptable execution time. In contrary to some sophisticated algorithms, ILS is a simple local search algorithm to be implemented whereas it produces promising performances in various kinds of combinational optimization problems (Lourenço, Martin et al. 2003, Stützle 2006, Pan and Ruiz 2012).

ILS starts with constructing an initial high-quality solution with effective heuristic, and later a local search is applied to improve the quality of this initial solution. Afterwards, a main loop, comprising perturbation, local search and acceptance criterion, is repeated until a termination criterion is satisfied. The perturbation aims at obtaining a new different solution to help the algorithm to escape from being trapped into local optima. The local search is utilized to improve this new solution, and the acceptance criterion is applied to determine whether this new solution replaces the incumbent one. The outline of the ILS algorithm is provided in Algorithm 1. From this procedure, it is clear that ILS algorithm is quite simple and quite easy for implementation. In fact, the main and the most complex part is the local search, and it is necessary to design an effective local search based on the characteristics of the considered problem. The solution presentation and the main segments of the ILS are presented in the following subsections.

Algorithm 1: Procedure of the ILS algorithm

$\pi_0 \leftarrow$ Obtain an initial high-quality solution;
$\pi \leftarrow$ Conduct local search on π_0; **%** Local search
Repeat
  $\pi' \leftarrow$ Conduct perturbation on π; **%** Perturbation
  $\pi'' \leftarrow$ Conduct local search on π'; **%** Local search
  % Acceptance criterion
  $\pi \leftarrow$ Utilize acceptance criterion to select one solution from π and π'';
Until termination criterion is satisfied
Output the best solution so far

3.1 Encoding and decoding

The proposed algorithm utilizes the task permutation for encoding following (Kalayci and Gupta 2013), (Kalayci and Gupta 2014). An encoding example of 4, 7, 8, 5, 6, 3, 1, 2 for the 8-part PC instance is illustrated in Figure 4. The task in the former position has higher priority and should be assigned at first, e.g. task 4 should be allocated at first. To transfer the encoding into a feasible solution, an effective decoding procedure is vital. However, the published decoding procedure for SDLBP cannot be applied to the SUDLBP due to the special precedence constraint. Hence, it is necessary to modify the decoding procedure in SDLBP to suit the SUDLBP. For better comparison, the decoding procedure for SDLBP is first presented in Algorithm 2 as follows, where U is the set of unallocated tasks and T_m is the completion time of current station.

Algorithm 2: Decoding procedure for SDLBP

Start

Step 1: When all tasks have been allocated, terminate this procedure; execute Step 2, otherwise.

Step 2: Open a new station for task assignment.

Step 3: Add the tasks, whose predecessors have been allocated, to available task set;

Step 4: Add the tasks to the assignable task set when cycle time constraint is satisfied.
% For task i, the cycle time is satisfied when $T_m + t_i + \max\{0, sd_{ji}\} \leq CT \; \exists j \in U$.

Step 5: If the assignable task set is empty, execute Step 1; execute Step 6, otherwise.

Step 6: Select the assignable task in the former position of the task permutation and allocate the selected task to the current station. Afterwards, go to Step 3.

End

The main procedure of the proposed decoding scheme for SUDLBP is presented in Algorithm 3, where U is the set of unallocated tasks. The sets A_{Ent} and A_{Ex} denote the allocated tasks to the entrance side and exit side, respectively. T_m is the completion time of the current station. Clearly, the decoding procedure for SUDLBP is much more complex than that for SDLBP. As seen, a task is available when all its predecessors or successors have been allocated. Both the available task set and the assignable task set need to be divided into two sub-sets: for the entrance side and the exit side. Regarding the cycle time constraint, both the unallocated tasks U and the allocated tasks on the exit side A_{Ex} should be considered as the product is disassembled from the entrance side to the exit side. Hence, the task operation sequence needs to be recoded to calculate the objective values.

Algorithm 3: Decoding procedure for SUDLBP

Start

Step 1: When all tasks have been allocated, terminate this procedure; execute Step 2, otherwise.

Step 2: Open a new station for task assignment.

Step 3: 1) Add the tasks, whose predecessors have been allocated to the entrance side, to the available task set AV_{Ent} on the entrance side;
2) Add the tasks, whose successors have been allocated to the exit side, to the available task set AV_{Ex} on the exit side;

Step 4: 1) Add the tasks in AV_{Ent} to the assignable task set AS_{Ent} on the entrance side when cycle time constraint is satisfied;
% For task i in AV_{Ent}, the cycle time is satisfied when $T_m + t_i + \max\{0, sd_{ji}\} \leq CT \; \exists j \in$

$A_{Ex} \cup UU$.

2) Add the tasks in AV_{Ex} to the assignable task set AS_{Ex} on the exit side when cycle time constraint is satisfied.

% For task i in AV_{Ex}, the cycle time is satisfied when $T_m + t_i + \max\{0, sd_{ji}\} \leq CT \; \exists j \in A_{Ex} \cup UU$.

Step 5: If both AS_{Ent} and AS_{Ex} are empty, execute Step 1; execute Step 6, otherwise.

Step 6: Select the assignable task in the former position of the task permutation from AS_{Ent} and AS_{Ex} and allocate the selected task in AS_{Ent} (or AS_{Ex}) to the entrance (or exit) side of the current station. Afterwards, go to Step 3.

End

Figure 4 illustrates the example solution presentation of the 8-part PC instance in Section 2.1. In this figure, the solution sequence or the task operation sequence is 1, 2, 3, 6, 5, 8, 7, 4 rather than the 4, 7, 8, 5, 6, 3, 1, 2 in the decoding since the AND predecessors of one task must be allocated to the former sub-stations (with larger index). On the basis of the solution sequence, the objective values are calculated in Table 2. In this table, the real removing time of task 6 is $t_6 + sd_{5,6} = 16 + 1$ and the real removing time of task 2 is $t_2 + sd_{3,2} = 10 + 4$ due to the sequence dependencies.

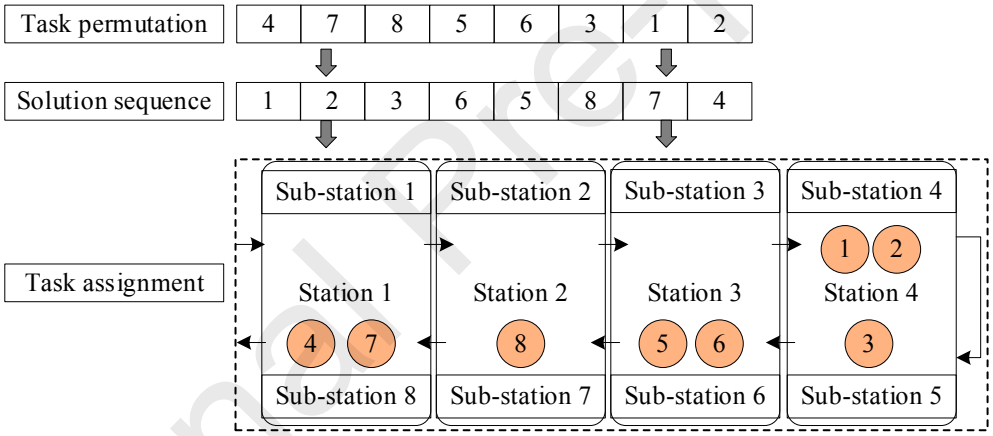


Figure 4 Solution presentation

Table 2 Calculation of the objectives

Stations	Sub-stations	Assigned tasks	Removing times of tasks	Total time	Idle time
Station 1	Sub-station 1	-	-	38	2
	Sub-station 8	7, 4	20, 18		
Station 2	Sub-station 2	-	-	36	4
	Sub-station 7	8	36		
Station 3	Sub-station 3	-	-	40	0
	Sub-station 6	6, 5	16+1, 23		
Station 4	Sub-station 4	1, 2	14, 10+4	40	0
	Sub-station 5	3	12		

$f_1 = 4$; $f_2 = 2^2 + 4^2 = 20$;

$f_3 = 1 \times 0 + 2 \times 0 + 3 \times 0 + 4 \times 0 + 5 \times 0 + 6 \times 0 + 7 \times 0 + 8 \times 0 = 0$;

$f_4 = 1 \times 360 + 2 \times 500 + 3 \times 620 + 4 \times 750 + 5 \times 540 + 6 \times 720 + 7 \times 295 + 8 \times 480 = 19145$;

3.2 Initialization with modified NEH heuristic

There are several simple heuristics to obtain initial solutions, such as the well-known ranked positional weight method. However, these heuristics might obtain poor solutions for some instances. Therefore, this study proposes a modified NEH heuristic based on the well-known NEH heuristic (Ruiz and Stützle 2007) to obtain a high-quality initial solution. The main procedure of the modified NEH heuristic is presented as follows.

Step 1: An initial task permutation π is obtained in the decreasing order of the ranked positional weights (the sum of the operation times of the task and all its successors).

Step 2: The task in the second position of the permutation is removed out and inserted into the first position, and the better one between the new solution and the incumbent solution is preserved.

Step 3: The task in the third position of the permutation is removed out and inserted into the first and the second positions, and the better one between the two new solutions and the incumbent solution is preserved. This procure is terminated when the task in the last position is removed out and inserted into all the former positions.

The main differences between the original NEH and the modified NEH lie in two aspects: (i) The original NEH obtains the initial task permutation based on the task operation times, whereas the modified NEH achieves the initial task permutation with the ranked positional weight method. (ii) The original NEH inserts the tasks into all the positions before its original position in the task permutation, and these new partial solutions are evaluated and the best partial solution is preserved. However, the partial solution is difficult to be evaluated in terms of the multiple objectives in this paper, and hence this modified NEH preserves the latter part of the task permutation to obtain a complete solution. As you will see in Section 4.3, this modified NEH produces superior performance than the simple heuristics.

3.3 Improved local search operator

Local search aims at finding the local optimal solution, which is the vital segment of the ILS algorithm. However, the effective local search operators in flowshop scheduling problem might not produce satisfying results observed in our preliminary experiments. Hence, this study produces a new local search with referenced permutation and two neighbor structures. To differentiate the proposed local search procedure from those published ones, this study exhibits two local search operators (Ruiz and Stützle 2007, Pan and Ruiz 2014).

The first local search applied in (Ruiz and Stützle 2007), referred to as OLS1, is illustrated in Algorithm 4, where π means the current solution. Within this local search procedure, a task is removed out from π and inserted into any position of the permutation, and the incumbent permutation is updated only when improvement is achieved. This procedure terminates when no improvement is achieved in N consecutive iterations. On the basis of this local search, (Pan and Ruiz 2014) developed an improved version, referred to as OLS2, by employing a *referenced permutation* which is the best task permutation found so far. Instead of removing one task randomly, this new local search removes the tasks in the task permutation in sequence. Hence, OLS2 ensures that all the tasks are selected in sequence and avoids selecting the same task again and again.

Algorithm 4: Procedure of OLS1 (π)

improve: = true;

Repeat

improve: = false;
For i:=1 **to** N **do**
Remove one task j from π randomly (without repetition);
$\pi'\leftarrow$ best task permutation by inserting task j into any position of permutation π (without repetition);
If $\mathrm{Fit}(\pi') < \mathrm{Fit}(\pi)$
$\pi: = \pi'$ and improve: = true;
Endif
Endfor
Until (improve = false**)**;

For the SUDLBP, OLS1 and OLS2 cannot show a satisfying performance due to the special characteristics of the SUDLBP. Hence, this research develops a new local search approach, referred to as LS, and the main procedure is illustrated in Algorithm 5. Here, π represents the current solution, π^{rp} is the best task permutation found so far, and a and b are two parameters. The main features of this local search are described as follows. (i) The utilization of π^{rp} makes sure that all tasks are selected in sequence and one-by-one. (ii) This study utilizes parameter a as the number of the neighbor solutions to speed up the search process. In fact, there are many identical solutions when inserting one task to all possible positions. (iii) The proposed local search utilizes both insert operator and swap operator to scan the search space efficiently. The algorithm is terminated when no improvement is achieved after executing $a \times b$ times insert or swap operator for each task.

To evaluate the performance of the proposed method, LS is compared with OLS1, OLS2 and two more versions: OLS3 where the referenced permutation is not employed in Algorithm 5 and OLS4 where only the insert operator is employed in Algorithm 5. As you will see in Section 4.3, the proposed LS outperforms the compared two ones, demonstrating the effectiveness of these improvements.

Algorithm 5: Procedure of local search LS (π,π^{rp}, a, b)

Set $counter$:=0 and i:=1;
Repeat
Remove task π_i^{rp} from π;
% π_i^{rp} is the task in the ith position of the best task permutation π^{rp};
For j:=1 **to** b **do**
If ($rand \leq 0.5$)
$\pi'\leftarrow$ Permutation by inserting task π_i^{rp} into a randomly selected position in π (without repetition);
Else
$\pi'\leftarrow$ Permutation by exchanging the positions of task π_i^{rp} and another randomly selected task in π (without repetition);
Endif
Set $counter$:=0 when $\mathrm{Fit}(\pi') < \mathrm{Fit}(\pi)$;
% $\mathrm{Fit}(\pi)$ is the fitness of solution π;
Replace π with π' when $\mathrm{Fit}(\pi') \leq \mathrm{Fit}(\pi)$;
$counter$:=$counter + 1$ and i:=$1 + \mathrm{mod}(\mathrm{i} + 1,\mathrm{N})$;
Endfor

Until $counter = a \cdot N$;

3.4 Perturbation and acceptance criterion

Perturbation operator aims at exploring new regions and increasing the exploration capacity by modifying the current task permutation. In the proposed ILS algorithm, perturbation obtains a number of neighbor solutions (set to 50) by executing a total of γ times (set to 6) of insert operator and swap operator. The values of parameter γ should be carefully determined and small value leads to difficulties in escaping from local optima and a large value leads to a completely new solution inheriting little information from the incumbent solution. Afterwards, the best one among the generated neighbor solutions is selected as the new solution and local search is conducted to improve this individual.

As for the newly achieved solution, it is necessary to determine whether this solution replaces the incumbent solution. While there are some studies to utilize the SA acceptance criterion to accept the worse solution with a certain probability (see (Pan and Ruiz 2012)). For simplicity, this study utilizes one simple acceptance criterion where the new solution replaces the incumbent one only when the same or better objective values are obtained.

4. Computational study and results

This section carries out the computational studies to test the proposed model and the algorithm. To have a better observation of the algorithms' performance, two sets of benchmarks are solved: the first set contains two different scale cases taken from the studies on the SDLBP and the second set contains 47 instances by (Kalayci, Polat et al. 2016). The number of tasks in the 47 instances ranges from a small number of 7 to a large number of 148. Specifically, Mukherjee has 94 tasks, Arcus2 has 111 tasks and Barthol2 has 148 tasks (see Section 4.3). Moreover, a set of eight algorithms are re-implemented their performances are compared with the proposed ILS algorithm. These algorithms are hill-climbing algorithm – HC (McGovern and Gupta 2007), late acceptance hill-climbing algorithm (Yuan, Zhang et al. 2015), simulated annealing algorithm (SA), tabu search algorithm (TS), genetic algorithm (GA), artificial bee colony algorithm (ABC), bees algorithm (BA) and particle swarm optimization algorithm (PSO). The main procedures and utilized parameters of the implemented algorithms are omitted for space reasons, but they are available upon request.

First of all, the solutions of two different scale cases are given in Section 4.1 and Section 4.2 to compare the performance of the U-shaped line and the straight line and test the performance of the proposed ILS algorithm. Subsequently, the improvements on the proposed algorithm are evaluated, where the performance of the modified NEH heuristic is compared with four simple heuristics and the performance of the local search is compared with four others. Afterwards, the performance of the proposed ILS is compared with the model utilizing CPLEX solver or MINLP solver and eight other implemented algorithms utilizing the first instance set (comparative study-1). Finally, the performance of the proposed ILS is compared with the hybrid genetic algorithm (Kalayci, Polat et al. 2016) and eight re-implemented algorithms utilizing the second instance set (comparative study-2).

Notice that, when solving the large-size instances, it is observed that the optimal station numbers cannot be achieved for some 'hard' instances. The reason lies behind is that the line balance optimization is the secondary objective and the solution with better line balance is preserved when utilizing the hierarchy method. However, in preliminary experiments, it is observed that the secondary objective, on the contrary, might be the obstacle to reduce the station number as it is difficult or even impossible to reduce the station number when the loads are balanced. Also, there is a higher probability of transferring a solution with less load on the last station into a solution with fewer stations with small modifications. Hence, this

study divides the algorithms into two phases when solving the large-size instances. Phase I utilizes the objective of minimizing the $\mathrm{ns} + (T_{ns-1} + T_{ns})/(2 \cdot CT)$, with the purpose of reducing the station number by preserving the solution with less loads on the last two stations. Phase II utilizes the objectives in Section 2.2 and further optimizes the latter three objectives on the basis of the solution achieved in Phase I. All algorithms terminate with a computation time limit of 1000s, where Phase I terminates when the optimal station number is achieved or the computation time reaches to the half of the time limit.

All the algorithms are executed for 20 times to solve each instance. The termination criterion is the CPU time of 100s for small-size instances with less than 70 tasks and 300s for large-size instances with 70 or more tasks. These algorithms are programmed in C++ language on an Intel Core i7-4790S 3.20 GHZ CPU 8.0 GB RAM personal computer. The models are solved utilizing General Algebraic Modeling System (GAMS) 23.0, and the MILP model is terminated when optimal solution is verified or the execution time reaches to 3600 s and the MINLP model is terminated when the non-linear programming (NLP) subproblems start to deteriorate or the execution time reaches to 3600 s.

4.1 Case study 1

The first instance contains 10 parts, referred to as P10, and it is taken from (Kalayci and Gupta 2013). The precedence diagram of the 10-part product is illustrated in Figure 5 and the database is provided in Table 3. The sequence dependencies of the 10-part instance are provided as follows: $sd_{1,4} = 1$, $sd_{4,1} = 4$; $sd_{2,3} = 2$, $sd_{3,2} = 3$; $sd_{4,5} = 4$, $sd_{5,4} = 2$; $sd_{5,6} = 2$, $sd_{6,5} = 4$; $sd_{6,9} = 3$, $sd_{9,6} = 1$.

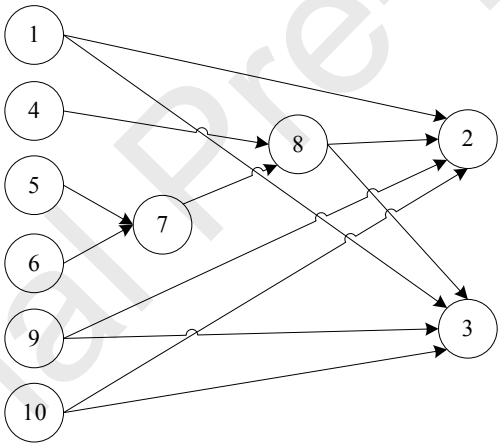


Figure 5 The precedence diagram of 10-part product by (Kalayci and Gupta 2013)

Table 3 Database of the 10-part product by (Kalayci and Gupta 2013)

Task	Part removal time	Hazardous	Demand
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 presents the best value (Best), average value (Avg) and the standard deviation (S.D.) of the results by the proposed ILS in solving SDLBP on the straight line and SUDLBP on the U-shaped line with a cycle time of 40. Figure 6 illustrates the near-optimal task assignment in the straight line and the

U-shaped line. From this table, it is observed that the U-shaped line obtains the smaller value of f_2, and it is sufficient to conclude that U-shaped line obtains better line balance since the objectives are considered hierarchically. ILS is capable of obtaining the current best results of SDLBP reported in (Liu and Wang 2017) at each run, which suggests that the proposed ILS is quite effective and robust. Figure 6 might provide the reasons leading to the superiority of the U-shaped line. In the U-shaped line, tasks can be allocated to the entrance side and exit side, whereas tasks in a straight line must be allocated to the entrance side (the straight line can be regarded as the U-shaped line with only entrance side). Clearly, U-shaped line has higher flexibility and thus obtains superior line balance.

Table 4 Results of P10 for straight line and U-shaped line

Layout	Criteria	f_1	f_2	f_3	f_4
Straight line	Best	5	67	5	9605
	Avg	5.00	67.00	5.00	9605.00
	S.D.	0.00	0.00	0.00	0.00
U-shaped line	Best	5	61	6	8880
	Avg	5.00	61.00	6.00	8880.00
	S.D.	0.00	0.00	0.00	0.00

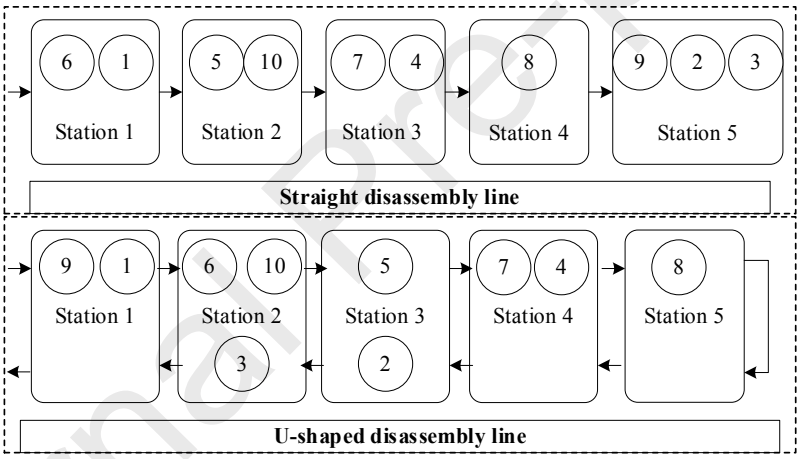


Figure 6 Task assignment of P10 for straight line and U-shaped line

4.2 Case study 2

The second case is a cellular telephone instance with 25 parts, referred to as P25 taken from (Kalayci and Gupta 2013). The precedence diagram and the database of the 25-part cellular telephone instance are given in Figure 7 and Table 5; respectively. The sequence dependencies of the cellular telephone instance are provided as follows: $sd_{4,5}=2, sd_{5,4}=1; sd_{6,7}=1,\ sd_{7,6}=2;\ sd_{6,9}=2,\ sd_{9,6}=1;\ sd_{7,8}=1, sd_{8,7}=2;\ sd_{13,14}=1, sd_{14,13}=2;\ sd_{14,15}=2, sd_{15,14}=1;\ sd_{20,21}=1, sd_{21,20}=2; sd_{22,25}=1, sd_{25,22}=2.$

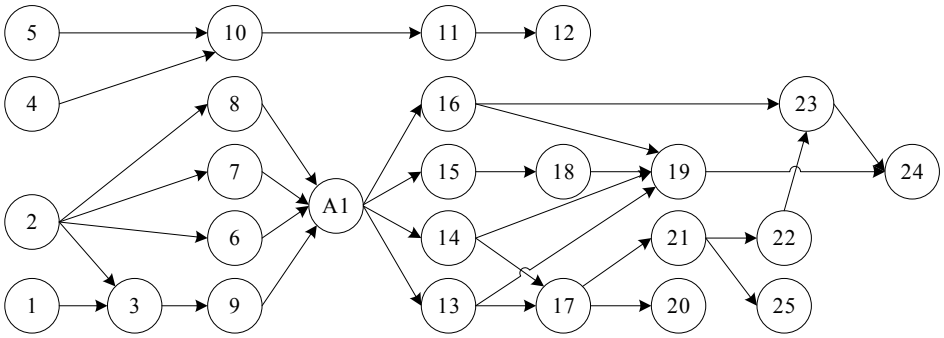


Figure 7 The precedence diagram of 25-part cellular telephone instance by (Kalayci and Gupta 2013)

Table 5 Database of the 25-part cellular telephone instance by (Kalayci and Gupta 2013)

Task	Part name	Part removal time	Hazardous	Demand
1	Antenna	3	Yes	4
2	Battery	2	Yes	7
3	Antenna guide	3	No	1
4	Bolt (Type 1) A	10	No	1
5	Bolt (Type1) B	10	No	1
6	Bolt (Type2) 1	15	No	1
7	Bolt (Type2) 2	15	No	1
8	Bolt (Type2) 3	15	No	1
9	Bolt (Type2) 4	15	No	1
10	Clip	2	No	2
11	Rubber seal	2	No	1
12	Speaker	2	Yes	4
13	White cable	2	No	1
14	Red/blue cable	2	No	1
15	Orange cable	2	No	1
16	Metal top	2	No	1
17	Front cover	2	No	2
18	Back cover	3	No	2
19	Circuit board	18	Yes	8
20	Plastic screen	5	No	1
21	Keyboard	1	No	4
22	LCD	5	No	6
23	Sub-keyboard	15	Yes	7
24	Internal IC board	2	No	1
25	Microphone	2	Yes	4

Table 6 illustrates the detailed results by ILS on both straight line and U-shaped line with a cycle time of 18. It is clear that the solutions for the straight line and the U-shaped line have the same values of f_1 and f_2, whereas U-shaped line has the smaller value of f_3. This finding suggests that the U-shaped line is capable of removing the hazardous parts at first. Again, the proposed ILS obtains the current best results reported in (Liu and Wang 2017) each time in solving the SDLBP, verifying the effectiveness of the proposed ILS algorithm once again.

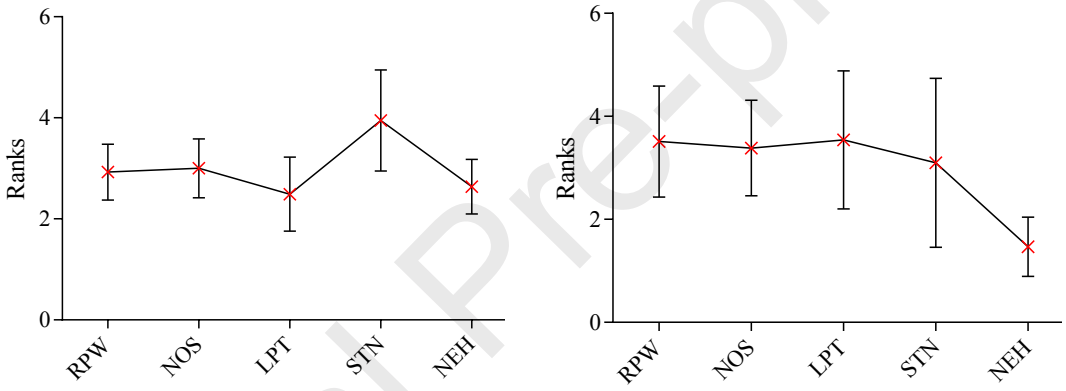
Table 6 Results of P25 for straight line and U-shaped line

Layout	Criteria	f_1	f_2	f_3	f_4
Straight line	Best	10	9	80	925
	Avg	10.00	9.00	80.00	925.00
	S.D.	0.00	0.00	0.00	0.00
U-shaped line	Best	10	9	76	909
	Avg	10.00	9.00	77.65	913.40
	S.D.	0.00	0.00	1.53	4.08

4.3 Evaluation of the improvements

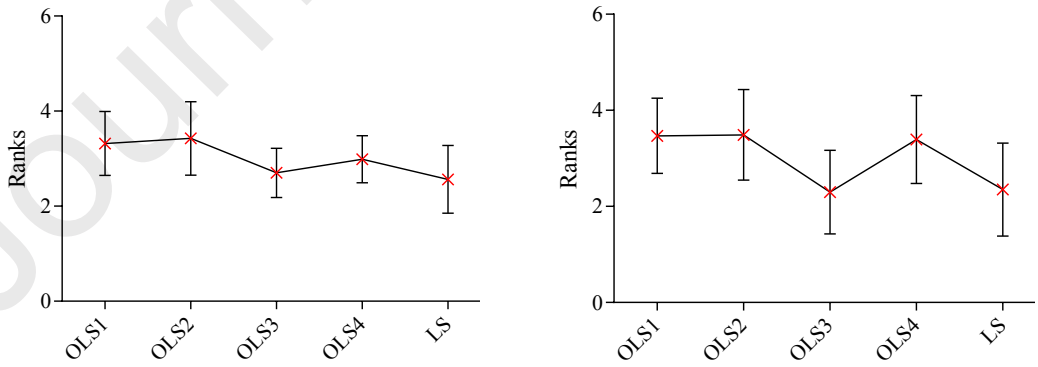
This section utilizes the second instance set to evaluate the improvements presented in Section 3. In order to evaluate the improvements statistically, the non-parametric Friedman rank-based analysis as the normality of the residuals is violated, where the obtained (average) result for each instance is regarded as the response variable. Supposed that there are five factors, the best result is provided with a rank of 1 and the worst result is given a rank of 5, and the factor with the smallest rank is regarded as the best performer.

The modified NEH heuristic is compared with four simple heuristics: ranked positional weight (RPW), number of successors (NOS), longest operation time (LPT) and smallest task number (STN). Figure 8 illustrates the average ranks regarding f_1 and f_2, where the Friedman rank-based analysis shows that there is a statistically significant difference between the factors in terms of f_1 or f_2. As you can see, the proposed NEH shows a similar performance to RPW, NOS and LPT and the better performance than STN in terms of f_1. Nevertheless, NEH outperforms other four simple heuristics by a significant margin in terms of f_2. In summary, this comparative study demonstrates that the proposed NEH has a superior performance with the cost of more computational effort.



(a) Ranks in terms of f_1 (b) Ranks in terms of f_2

Figure 8 Means plot of the average ranks and 95% confidence intervals of the heuristics



(a) Ranks in terms of f_1 (b) Ranks in terms of f_2

Figure 9 Means plot of the average ranks and 95% confidence intervals of the local search operators

Figure 9 presents the ranks in terms of f_1 and f_2 by ILS with different local search operators in Section 3.3, where all the algorithms are executed for 20 times. Notice that, as this study utilizes the hierarchy method in (McGovern and Gupta 2006) and these local search operators have shown clear difference,

the results in terms of other objectives are omitted for space reasons, but they are available upon request. As you can see, LS outperforms the published two local search methods: OLS1 and OLS2, demonstrating the effectiveness of the improvements. Also, LS outperforms OLS4 clearly, proving that the utilization of two neighbor operators is efficient.

4.4 Comparative study-1

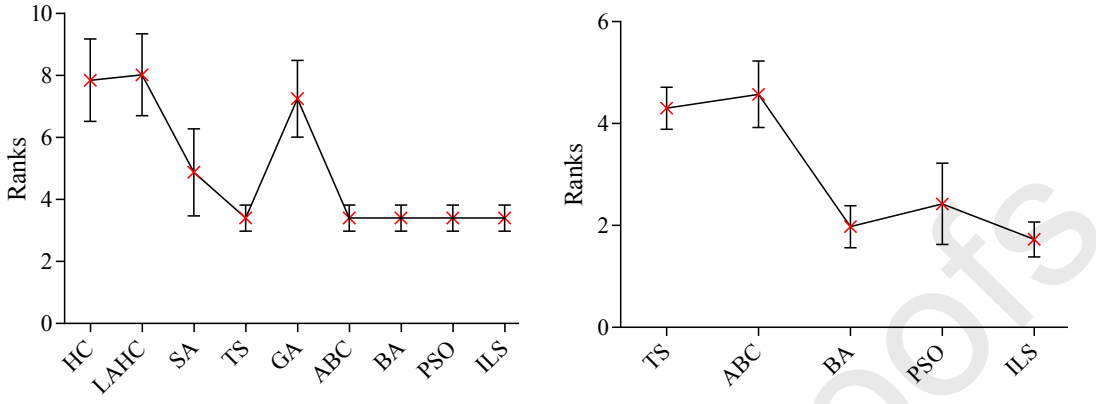
This section illustrates the detailed results by all the implemented algorithms for the first instance set in Table 7. From this table, it is observed that ILS, SA, TS, GA, ABC, BA and PSO are seven best performers in solving P10, and they outperform all the other algorithms regarding by means of f_2. The local search algorithm, HC and LAHC, show the worst performance due to being trapped into local optima. Regarding P25, ILS obtains the best results each time with a standard deviation of 0.0. Specifically, ILS outperforms HC, LAHC, SA and GA in terms of f_2 and outperforms all the other algorithms in terms of f_3. In order to evaluate the algorithms statistically, this study also carries out the non-parametric Friedman rank-based analysis, where the obtained result in each run is regarded as the response variable. The average ranks regarding f_2 and f_3 in solving P25 are illustrated in Figure 10 as the algorithms achieve the same results in terms of f_1. The statistical analysis suggests that there is a statistically significant difference (see Figures 10(a) and 10(b)) and the proposed ILS outperforms HC, LAHC, SA and GA statistically in terms of f_2 and TS and ABC statistically in terms of f_3. The comparative results show that ILS outperform all the algorithms or obtain the same results when solving P10 and P25, which demonstrates that ILS is quite effective and efficient for the SUDLBP.

Table 7 Results by implemented algorithms

Instances	Objectives	Criteria	HC	LAHC	SA	TS	GA	ABC	BA	PSO	ILS
P10		Best	5	5	5	5	5	5	5	5	5
	f_1	Avg	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
		S.D.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		Best	61	61	61	61	61	61	61	61	61
	f_2	Avg	63.35	62.60	61.00	61.00	61.00	61.00	61.00	61.00	61.00
		S.D.	5.30	2.80	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		Best	6	4	6	6	6	6	6	6	6
	f_3	Avg	6.25	6.10	6.00	6.00	6.00	6.00	6.00	6.00	6.00
		S.D.	0.43	0.62	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		Best	8880	8590	8880	8880	8880	8880	8880	8880	8880
	f_4	Avg	9425.75	9259.00	8880.00	8880.00	8880.00	8880.00	8880.00	8880.00	8880.00
		S.D.	945.68	809.86	0.00	0.00	0.00	0.00	0.00	0.00	0.00
P25		Best	10	10	10	10	10	10	10	10	10
	f_1	Avg	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00	10.00
		S.D.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		Best	9	9	9	9	9	9	9	9	9
	f_2	Avg	13.70	13.80	10.05	9.00	12.20	9.00	9.00	9.00	9.00
		S.D.	2.19	2.50	1.20	0.00	1.60	0.00	0.00	0.00	0.00
		Best	75	75	76	77	75	80	76	76	76
	f_3	Avg	78.95	80.30	80.15	80.95	80.35	81.70	76.40	77.25	76.00
		S.D.	3.54	2.90	4.84	2.94	2.15	1.55	0.92	2.28	0.00
		Best	873	872	884	916	845	925	909	910	909
	f_4	Avg	892.15	887.40	906.00	929.30	886.05	932.95	911.35	913.80	909.00
		S.D.	15.69	16.20	8.93	7.80	16.78	5.92	3.89	5.98	0.00

Table 8 presents the computational results by the model and ILS algorithm to solve the two models with different objectives, where N/A means no integer solution found within the time limit given. The first model is a MILP model to minimize f_1; the second model is a MINLP to minimize $100 \times f_1 + f_2$. From this table, it is observed that CPLEX and ILS obtain the same value of f_1, whereas CPLEX costs a large amount of time (3600s) when solving P25. Regarding the minimization of $100 \times f_1 + f_2$, the MINLP model achieves worse results than ILS when solving P10 and P25 and terminates as the objective function

of the NLP subproblems start to deteriorate. As f_1 and f_2 have higher priorities and the results are capable of differentiating the performances of the developed model and ILS algorithm, the objective f_3 and objective f_4 are not tested here for simplicity.



(a) Ranks in terms of f_2 (b) Ranks in terms of f_3

Figure 10 Means plot of the average ranks and 95% confidence intervals of algorithms regarding f_2 and f_3 when solving the P25

Table 8 Results by the developed model and ILS algorithm

Objectives	Instance	Method	f_1	f_2	Time (s)
Min f_1	P8	CPLEX	4	-	0.555
		ILS	4	-	-
	P10	CPLEX	5	-	0.701
		ILS	5	-	-
	P25	CPLEX	10	-	3600
		ILS	10	-	-
Min $100 \times f_1 + f_2$	P8	MINLP solver	4	20	2.735*
		ILS	4	20	-
	P10	MINLP solver	5	69	35.637*
		ILS	5	67	-
	P25	MINLP solver	N/A	N/A	3600
		ILS	10	9	-

**Terminated due to that the objective function of the NLP subproblems started to deteriorate*

In summary, the two case studies demonstrate that the U-shaped line is capable of obtaining a better line balance than traditional straight line due to higher flexibility. The comparative study shows that the proposed ILS outperforms five other implemented algorithms and the developed models, demonstrating that ILS is quite effective for the SUDLBP despite being simple.

4.5 Comparative study-2

This section presents the computational results for the second instance set. Table 9 reports the best results of SDLBP by genetic algorithm with a variable neighborhood search method (VNSGA) (Kalayci, Polat et al. 2016), the best results of SDLBP by the proposed ILS method and the best results of SUDLBP by the proposed ILS method in terms of f_1 and f_2. Notice that the results by VNSGA is taken from

(Kalayci, Polat et al. 2016) directly and the results in terms of other objectives are available upon request. It is observed that ILS obtains respectively better and the same results for 7 and 40 instances in term of f_1 compared with VNSGA. ILS obtains better, the same and worse results for 20, 10 and 17 instances, respectively, in term of f_2. As this study utilizes the hierarchy method for performance evaluation, it might be concluded that the proposed ILS performs superior performance. It is also observed that the U-shaped line is capable of obtaining better line balance than the traditional straight line for most instances. Specifically, U-shaped line utilizes less station number (f_1) for 14 cases and obtains better line balance (f_2) for 40 cases.

Table 9 Comparison between VNSGA and ILS

Instances	CT	Results of SDLBP by VNSGA		Results of SDLBP by ILS		Results of SUDLBP by ILS	
		f_1	f_2	f_1	f_2	f_1	f_2
Mertens	7	5	10	5	10	5	10
Bowman	20	5	149	5	149	4	13
Jaeschke	7	7	26	7	28	7	28
Jackson	10	5	6	5	6	5	4
Mansoor	94	2	5	2	5	2	5
Mitchell	15	8	31	8	43	8	29
Roszieg	16	8	5	8	5	8	3
Heskiaoff	216	5	628	5	630	5	628
Buxey	30	12	118	12	122	11	6
Lutz1	2,357	7	8.13E+05	7	8.47E+05	7	7.99E+05
Gunther	41	14	1519	14	1735	12	13
Kilbridge	62	9	6	9	6	9	6
Hahn	2,806	6	1.87E+06	6	1.91E+06	5	6
Tonge	168	22	2152	22	1756	22	1672
	170	22	3002	22	2660	21	204
	173	22	5196	21	1081	21	745
	179	21	3459	20	312	20	262
	182	20	968	20	912	20	854
Wee-Mag	46	35	983	34	399	34	349
	47	33	148	33	116	33	106
	49	32	189	32	163	32	155
	50	32	347	32	333	32	327
	52	31	455	31	443	31	431
Arcus1	3,985	20	9.34E+05	20	9.22E+05	20	8.14E+05
	5,048	16	1.76E+06	16	1.76E+06	16	1.67E+06
	5,853	14	2.79E+06	14	2.79E+06	13	1.16E+04
	6,842	12	4.26E+06	12	4.25E+06	12	3.43E+06
	7,571	11	5.37E+06	11	5.54E+06	11	5.37E+06
	8,412	10	7.09E+06	10	7.83E+06	10	7.93E+06
	8,898	9	2.14E+06	9	2.15E+06	9	2.13E+06
	10,816	8	1.49E+07	8	3.75E+07	7	1.10E+01
Lutz2	15	34	63	34	61	33	10
Lutz3	150	12	2050	12	2256	11	6
Mukherjee	201	23	12057	23	14853	21	13
	301	15	10137	15	10137	14	6
Arcus2	5,755	27	2.58E+06	27	2.40E+06	27	1.06E+06
	7,520	21	3.00E+06	21	2.97E+06	21	2.75E+06
	8,847	18	4.38E+06	18	4.59E+06	18	4.41E+06
	10,027	16	6.33E+06	16	6.39E+06	16	6.42E+06
	10,743	15	7.76E+06	15	7.82E+06	15	7.81E+06
	11,378	14	5.76E+06	14	5.72E+06	14	5.68E+06
	11,570	14	9.86E+06	14	1.02E+07	14	9.63E+06
	17,067	9	1.14E+06	9	1.14E+06	9	1.14E+06
Barthol2	85	52	906	51	293	51	243
	89	50	1174	49	425	48	74
	91	49	1179	48	504	47	67
	95	47	1279	46	454	45	53

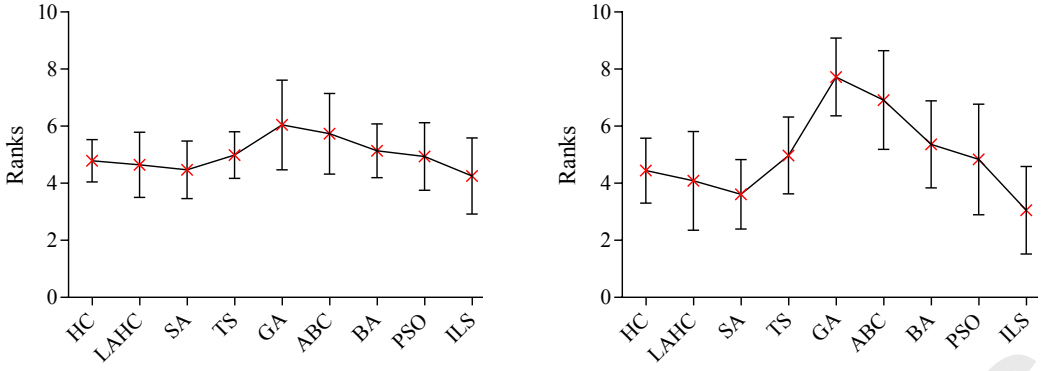
Table 10 and Table 11 present the results of SUDLBP by the nine implemented algorithms in terms of f_1 and f_2; respectively. Note that the detailed results for other objectives are also available upon request. From Table 10, it is observed that ILS obtains the peak performance when solving the largest-size

instance Barthol2. This study also conducts the Friedman rank-based analysis to evaluate these algorithms, and the statistical analysis shows that there is a statistically significant difference between the performances of the algorithms. Figure 11 illustrates the ranks in term of f_1, where Figure 11(a) presents the average ranks for all the instances and Figure 11(b) presents the average ranks for the instance which algorithms show different performance to highlight the difference. From the Friedman rank-based analysis, it is observed that ILS is the best performer in terms of f_1, and it outperforms TS, GA, ABC and BA by a significant margin.

Table 10 Results of SUDLBP in term of f_1 by nine implemented algorithms

Instances	CT	HC	LAHC	SA	TS	GA	ABC	BA	PSO	ILS
Mertens	7	5	5	5	5	5	5	5	5	5
Bowman	20	4	4	4	4	4	4	4	4	4
Jaeschke	7	7	7	7	7	7	7	7	7	7
Jackson	10	5	5	5	5	5	5	5	5	5
Mansoor	94	2	2	2	2	2	2	2	2	2
Mitchell	15	8	8	8	8	8	8	8	8	8
Roszieg	16	8	8	8	8	8	8	8	8	8
Heskiaoff	216	5	5	5	5	5	5	5	5	5
Buxey	30	11	11.05	11	11	11	11	11	11	11
Lutz1	2,357	7	7	7	7	7	7	7	7	7
Gunther	41	12	12	12	12	12.15	12	12	12	12
Kilbridge	62	9	9	9	9	9	9	9	9	9
Hahn	2,806	5.7	5.65	5.6	5.55	5.9	5.85	5.8	6	**5.2**
Tonge	168	22	22	22	22	22	22	22	21.85	22
	170	21.95	21.95	21.95	21.95	22	22	21.9	**21**	21.8
	173	21	21	21	21	21.3	21	21	21	21
	179	20	20	20	20	20.5	20	20	20	20
	182	20	20	20	20	20	20	20	20	20
Wee-Mag	46	34	34	34	34	34.3	34.95	34	34	34
	47	33	33	33	33	33	33	33	33	33
	49	32	32	32	32	32	32	32	32	32
	50	32	32	32	32	32	32	32	32	32
	52	31	31	31	31	31	31	31	31	31
Arcus1	3,985	20	20	20	20	20	20	20	20	20
	5,048	16	16	16	16	16	16	16	16	16
	5,853	13	13	13	13	13.7	14	13.85	13.4	13
	6,842	12	12	12	12	12	12	12	12	12
	7,571	11	11	11	11	11	11	11	11	11
	8,412	10	10	10	10	10	10	10	10	10
	8,898	9	9	9	9	9	9	9	9	9
	10,816	8	8	7.95	8	8	8	8	8	7.8
Lutz2	15	33	33	33	33	33.25	33.15	33	33	33
Lutz3	150	11	11	11	11	11.4	11.25	11	11	11
Mukherjee	201	21.25	21.2	**21.05**	21.95	22	22	22	21.85	21.25
	301	14	14	14	14.1	14.9	15	14.85	14.25	14
Arcus2	5,755	27	27	27	27	27	27	27	27	27
	7,520	21	21	21	21	21	21	21	21	21
	8,847	18	18	18	18	18	18	18	18	18
	10,027	16	16	16	16	16	16	16	16	16
	10,743	15	15	15	15	15	15	15	15	15
	11,378	14	14	14	14	14	14	14	14	14
	11,570	14	14	14	14	14	14	14	14	14
	17,067	9	9	9	9	9	9	9	9	9
Barthol2	85	51	51	51	51.05	51.9	51.95	51	51	51
	89	49	48.9	48.95	49	49.15	49	49	49	**48.75**
	91	48	47.8	47.7	48	48	48	48	48	**47.6**
	95	45.9	45.85	45.95	46	46	46	46	46	**45.65**

**Best in bold*



(a) Ranks in term of f_1 for all instances

(b) Ranks in term of f_1 for the instances where algorithms show different performance

Figure 11 Means plot of the average ranks and 95% confidence intervals of the tested algorithm

Table 11 Results of SUDLBP in term of f_2 by nine implemented algorithms

Instances	CT	HC	LAHC	SA	TS	GA	ABC	BA	PSO	ILS
Mertens	7	10	10	10	10	10	10	10	10	10
Bowman	20	13	13	13	13	13	13	13	13	13
Jaeschke	7	28	28	28	28	28	28	28	28	28
Jackson	10	4	4	4	4	4	4	4	4	4
Mansoor	94	5	5	5	5	5	5	5	5	5
Mitchell	15	30.7	31	29.2	30.2	29.7	29	29	30.7	29.1
Roszieg	16	3.2	3.9	3	3	3	3	3	3	3
Heskiaoff	216	634.8	636.4	628.5	631	629.1	628	628	628.4	629.1
Buxey	30	8.4	15.8	9.1	7.7	6.9	6.2	6	6	6.5
Lutz1	2,357	838157	830279	822809	818854	809410	803854	803083	884961	804475
Gunther	41	13	13.4	13	13	55.35	13	13	12.9	13.1
Kilbridge	62	6.2	8.9	6.3	6	6	6	6	6	6
Hahn	2,806	1E+06	1E+06	947087	983009	1E+06	1E+06	1E+06	2E+06	344411
Tonge	168	1805.5	1811.3	3327.5	1716.4	1764.9	1855.9	1701.7	1549.9	1783
	170	2690.9	2651.8	3222.4	2443.9	2617	2728.7	2317.1	230.8	2159.8
	173	1088.8	1719.7	947.8	800.7	1950.1	1180.5	804.8	830.9	954.1
	179	325.6	518.5	291.8	282.7	1848.7	423.3	269.5	280.3	290.8
	182	934	1685.7	887.6	882.9	915.1	1024.4	879.9	896.8	879.9
Wee-Mag	46	475.4	457.5	365	375.7	603.6	1174.4	373	404.1	426.7
	47	128.5	118	107.3	109.5	114.8	141.9	106.2	129.3	117.3
	49	159.9	159.5	156.4	156.1	159.8	182.2	156.4	176.5	159.3
	50	337.8	331.5	326.4	326.7	331.5	361.4	327.6	363.4	330.5
	52	446.9	444.4	429.8	431.2	439.8	500.2	431.1	474.5	437.8
Arcus1	3,985	838896	835347	830507	815064	843913	914218	821147	907529	827898
	5,048	2E+06	2E+06	2E+06	2E+06	2E+06	2E+06	2E+06	2E+06	2E+06
	5,853	13515	19389	12100	12551	2E+06	3E+06	2E+06	1E+06	12786
	6,842	4E+06	4E+06	3E+06	3E+06	3E+06	4E+06	3E+06	3E+06	4E+06
	7,571	6E+06	6E+06	6E+06	6E+06	6E+06	6E+06	5E+06	8E+06	6E+06
	8,412	1E+07	1E+07	9E+06	9E+06	9E+06	8E+06	8E+06	1E+07	1E+07
	8,898	2E+06	2E+06	2E+06	2E+06	2E+06	2E+06	2E+06	2E+06	2E+06
	10,816	4E+07	4E+07	3E+07	3E+07	4E+07	3E+07	3E+07	5E+07	3E+07
Lutz2	15	10.3	16.5	10	10.3	15.05	16.65	10.1	10	10.1
Lutz3	150	6.4	10.7	6	6.7	846.2	545.9	6.1	7.3	6.6
Mukherjee	201	588.25	475.1	121.65	2049.2	2231	2377.9	2155.8	2394.4	564.35
	301	14.4	16.5	16.2	646.8	5881	6878.6	5468.2	2253.8	9.6
Arcus2	5,755	1E+06	2E+06	1E+06	1E+06	1E+06	1E+06	1E+06	1E+06	1E+06
	7,520	3E+06	3E+06	3E+06	3E+06	3E+06	3E+06	3E+06	4E+06	3E+06
	8,847	5E+06	5E+06	5E+06	5E+06	5E+06	5E+06	5E+06	6E+06	5E+06
	10,027	7E+06	7E+06	7E+06	7E+06	7E+06	7E+06	6E+06	9E+06	7E+06
	10,743	8E+06	8E+06	8E+06	8E+06	8E+06	8E+06	8E+06	1E+07	8E+06
	11,378	6E+06	6E+06	6E+06	6E+06	6E+06	6E+06	6E+06	6E+06	6E+06
	11,570	1E+07	1E+07	1E+07	1E+07	1E+07	1E+07	1E+07	1E+07	1E+07
	17,067	1E+06	1E+06	1E+06	1E+06	1E+06	1E+06	1E+06	1E+06	1E+06
Barthol2	85	259.8	258.4	720.8	276.45	732.9	917.35	259.1	303.6	257.4
	89	371.2	346	1045.4	371.1	520.25	704.6	370.1	430.4	294.65
	91	414	362.4	678.2	415.5	479.4	727.8	413.2	480	281.3
	95	419.4	396.95	1271.1	446	518.3	698.6	433.7	502.6	311.65

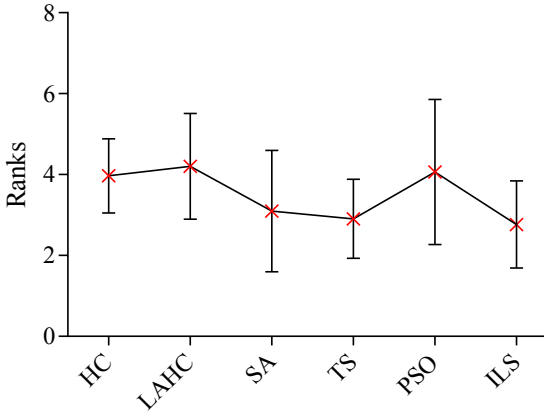


Figure 12 Means plot of the average ranks and 95% confidence intervals of the tested algorithm in term of f_2

From Table 11, it is observed that ILS also obtains the peak performance in term of f_2 when solving the largest-size instance Barthol2. This section also conducts the Friedman rank-based analysis to evaluate these algorithms and there is a statistically significant difference between the performances of the algorithms. Figure 12 presents the average ranks in term of f_2, where the six best performers in Figure 11 are tested. Recall that the hierarchy method is applied when evaluating the objectives and ILS has shown a superior performance in term of f_1 over the remaining ones. Clearly, the Friedman rank-based analysis suggests that the proposed ILS achieves the best performance. In summary, this comparative study demonstrates that the proposed algorithm achieves a competing performance in comparison with the other eight re-implemented algorithms.

5. Conclusions and future research

Disassembly lines are widely utilized in disassembling end-of-life products effectively, where sequence-dependent time increments occur in real-world applications. This research provides the first study to solve the multi-objective sequence-dependent disassembly line balancing problem in U-shaped lines, referred to as sequence-dependent U-shaped disassembly line balancing problem (SUDLBP). A mixed-integer programming model is developed to formulate the SUDLBP. The proposed model utilizes new expressions to deal with the AND/OR precedence relations by dividing one station into two sub-stations. It is capable of solving the small-size instances optimally. Due to the NP-hard structure of the considered problem, a simple and effective iterated local search (ILS) algorithm is developed to tackle the SUDLBP. The proposed ILS algorithm utilizes a new decoding procedure to obtain a feasible solution, where the assigned task set is divided into two sub-sets on the entrance side and the exit side. It also utilizes the modified NEH heuristic to achieve a high-quality initial solution. A new local search procedure with referenced permutation and two neighbor structures is also employed to emphasize intensification. Computational results show that the U-shaped layout provides advantages over the traditional straight lines to maximize line efficiency. The proposed ILS achieves competing performance in comparison with the eight re-implemented algorithms in the comparative study on two sets of instances. ILS also outperforms the developed MILP model in search speed and outperforms the MINLP model in terms of the search speed and the solution quality.

The methodology proposed in this research can easily be adopted by practitioners to have an efficient disassembly line system. The solution algorithm is easy to implement and can be used for balancing existing U-shaped disassembly lines. In the case that the problem size is very large in the real case applications (with hundreds of tasks), the number of neighbor solutions and insert-swap operators may

be increased to escape the local optima. Due to the superiority of the U-shaped layout over the straight line configuration (as shown in the computational study in this research), straight lines may also be converted to a U-shaped line and ILS can be used again to optimize the task assignments. However, this may have some practical difficulties as it may not be possible due to the restrictions caused by some organizational and/or technological circumstances.

Future research stems from applying this simple and effective ILS algorithm to other disassembly line balancing problems to study more realistic SUDLBPs. It is also suggested to study the stochastic SUDLBP, the fuzzy SUDLBP or other uncertainties in real-world applications. Money values for parts and some cost functions may also be included in the model with the aim of maximizing the total revenue. It is also interesting to develop some Pareto algorithms (Zhang, Wang et al. 2017, Zhu, Zhang et al. 2018) to solve multi-objective SUDLBP, and develop some exact methods, such as branch and bound algorithm (Li, Kucukkoc et al. 2018), to solve the large-size instances optimally.

References

Agrawal, S. and M. K. Tiwari (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): 1405-1429.

Altekin, F. T. and C. Akkan (2012). "Task-failure-driven rebalancing of disassembly lines." International Journal of Production Research 50(18): 4955-4976.

Altekin, F. T., L. Kandiller and N. E. Ozdemirel (2008). "Profit-oriented disassembly-line balancing." International Journal of Production Research 46(10): 2675-2693.

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

Avikal, S. and P. Mishra (2012). "A new U-shaped heuristic for disassembly line balancing problems." Internaional Journal of Science 1(1): 2277-7261.

Ding, L.-P., Y.-X. Feng, J.-R. Tan and Y.-C. Gao (2010). "A new multi-objective ant colony algorithm for solving the disassembly line balancing problem." The International Journal of Advanced Manufacturing Technology 48(5): 761-771.

Gharaei, A., S. A. Hoseini Shekarabi and M. Karimi (2019). "Modelling And optimal lot-sizing of the replenishments in constrained, multi-product and bi-objective EPQ models with defective products: Generalised Cross Decomposition." International Journal of Systems Science: Operations & Logistics: 1-13.

Gharaei, A., M. Karimi and S. A. Hoseini Shekarabi (2019). "An integrated multi-product, multi-buyer supply chain under penalty, green, and quality control polices and a vendor managed inventory with consignment stock agreement: The outer approximation with equality relaxation and augmented penalty algorithm." Applied Mathematical Modelling 69: 223-254.

Gharaei, A., M. Karimi and S. A. Hoseini Shekarabi (2019). "Joint Economic Lot-sizing in Multi-product Multi-level Integrated Supply Chains: Generalized Benders Decomposition." International Journal of Systems Science: Operations & Logistics: 1-17.

Gungor, A. and S. M. Gupta (2001). "A solution approach to the disassembly line balancing problem in the presence of task failures." International Journal of Production Research 39(7): 1427-1467.

Güngör, A. and S. M. Gupta (1999). Disassembly Line Balancing. Proceedings of the Annual Meeting of the Northeast Decision Sciences Institute, Newport, RI.

Güngör, A. and S. M. Gupta (2002). "Disassembly line in product recovery." International Journal of Production Research 40(11): 2569-2589.

Hoseini Shekarabi, S. A., A. Gharaei and M. Karimi (2019). "Modelling and optimal lot-sizing of integrated multi-level multi-wholesaler supply chains under the shortage and limited warehouse space: generalised outer approximation." International Journal of Systems Science: Operations & Logistics 6(3): 237-257.

Kalayci, C. B. and S. M. Gupta (2013). "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): 197-209.

Kalayci, C. B. and S. M. Gupta (2013). "Ant colony optimization for sequence-dependent disassembly line balancing problem." Journal of Manufacturing Technology Management 24(3): 413-427.

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

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

Kalayci, C. B., A. Hancilar, A. Gungor and S. M. Gupta (2015). "Multi-objective fuzzy disassembly line balancing using a hybrid discrete artificial bee colony algorithm." Journal of Manufacturing Systems 37: 672-682.

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

Kazemi, N., S. H. Abdul-Rashid, R. A. R. Ghazilla, E. Shekarian and S. Zanoni (2018). "Economic order quantity models for items with imperfect quality and emission considerations." International Journal of Systems Science: Operations & Logistics 5(2): 99-115.

Koc, A., I. Sabuncuoglu and E. Erel (2009). "Two exact formulations for disassembly line balancing problems with task precedence diagram construction using an AND/OR graph." IIE Transactions 41(10): 866-881.

Li, Z., I. Kucukkoc and Z. Zhang (2018). "Branch, bound and remember algorithm for U-shaped assembly line balancing problem." Computers & Industrial Engineering 124: 24-35.

Liu, J. and S. Wang (2017). "Balancing Disassembly Line in Product Recovery to Promote the Coordinated Development of Economy and Environment." Sustainability 9(2): 309.

Liu, J., Z. Zhou, D. T. Pham, W. Xu, J. Yan, A. Liu, C. Ji and Q. Liu (2018). "An improved multi-objective discrete bees algorithm for robotic disassembly line balancing problem in remanufacturing." International Journal of Advanced Manufacturing Technology 97(9-12): 3937-3962.

Lourenço, H. R., O. C. Martin and T. Stützle (2003). Iterated Local Search. Handbook of Metaheuristics. F. Glover and G. A. Kochenberger. Boston, MA, Springer US: 320-353.

McGovern, S. M. and S. M. Gupta (2003). 2-opt heuristic for the disassembly line balancing problem. Proceedings of the SPIE International Conference on Environmentally Conscious Manufacturing III, Providence, RI.

McGovern, S. M. and S. M. Gupta (2006). "Ant colony optimization for disassembly sequencing with multiple objectives." The International Journal of Advanced Manufacturing Technology 30(5): 481-496.

McGovern, S. M. and S. M. Gupta (2007). "A balancing method and genetic algorithm for disassembly line balancing." European Journal of Operational Research 179(3): 692-708.

McGovern, S. M. and S. M. Gupta (2007). "Combinatorial optimization analysis of the unary NP-complete disassembly line balancing problem." International Journal of Production Research 45(18-19): 4485-4511.

Mete, S., Z. A. Çil, K. Ağpak, E. Özceylan and A. Dolgui (2016). "A solution approach based on beam search algorithm for disassembly line balancing problem." Journal of Manufacturing Systems 41: 188-200.

Mete, S., Z. A. Çil, E. Özceylan, K. Ağpak and O. Battaïa (2018). "An optimisation support for the design of hybrid production lines including assembly and disassembly tasks." International Journal of Production

Research 56(24): 7375-7389.

Paksoy, T., A. Güngör, E. Özceylan and A. Hancilar (2013). "Mixed model disassembly line balancing problem with fuzzy goals." International Journal of Production Research 51(20): 6082-6096.

Pan, Q.-K. and R. Ruiz (2012). "Local search methods for the flowshop scheduling problem with flowtime minimization." European Journal of Operational Research 222(1): 31-43.

Pan, Q.-K. and R. Ruiz (2014). "An effective iterated greedy algorithm for the mixed no-idle permutation flowshop scheduling problem." Omega 44: 41-50.

Rabbani, M., N. Foroozesh, S. M. Mousavi and H. Farrokhi-Asl (2019). "Sustainable supplier selection by a new decision model based on interval-valued fuzzy sets and possibilistic statistical reference point systems under uncertainty." International Journal of Systems Science: Operations & Logistics 6(2): 162-178.

Rabbani, M., S. A. A. Hosseini-Mokhallesun, A. H. Ordibazar and H. Farrokhi-Asl (2018). "A hybrid robust possibilistic approach for a sustainable supply chain location-allocation network design." International Journal of Systems Science: Operations & Logistics: 1-16.

Ren, Y., D. Yu, C. Zhang, G. Tian, L. Meng and X. Zhou (2017). "An improved gravitational search algorithm for profit-oriented partial disassembly line balancing problem." International Journal of Production Research 55(24): 7302-7316.

Ren, Y., C. Zhang, F. Zhao, G. Tian, W. Lin, L. Meng and H. Li (2018). "Disassembly line balancing problem using interdependent weights-based multi-criteria decision making and 2-Optimal algorithm." Journal of Cleaner Production 174: 1475-1486.

Ren, Y., C. Zhang, F. Zhao, M. J. Triebe and L. Meng (2018). "An MCDM-Based Multiobjective General Variable Neighborhood Search Approach for Disassembly Line Balancing Problem." IEEE Transactions on Systems, Man, and Cybernetics: Systems: 1-14.

Ren, Y., C. Zhang, F. Zhao, H. Xiao and G. Tian (2018). "An asynchronous parallel disassembly planning based on genetic algorithm." European Journal of Operational Research 269(2): 647-660.

Ruiz, R. and T. Stützle (2007). "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem." European Journal of Operational Research 177(3): 2033-2049.

Stützle, T. (2006). "Iterated local search for the quadratic assignment problem." European Journal of Operational Research 174(3): 1519-1539.

Tsao, Y.-C. (2015). "Design of a carbon-efficient supply-chain network under trade credits." International Journal of Systems Science: Operations & Logistics 2(3): 177-186.

Urban, T. L. (1998). "Note. Optimal Balancing of U-Shaped Assembly Lines." Management Science 44(5): 738-741.

Urban, T. L. and W.-C. Chiang (2006). "An optimal piecewise-linear program for the U-line balancing problem with stochastic task times." European Journal of Operational Research 168(3): 771-782.

Wang, S., X. Guo and J. Liu (2019). "An efficient hybrid artificial bee colony algorithm for disassembly line balancing problem with sequence-dependent part removal times." Engineering Optimization: 1-18.

Xiao, S., Y. Wang, H. Yu and S. Nie (2017). "An Entropy-Based Adaptive Hybrid Particle Swarm Optimization for Disassembly Line Balancing Problems." Entropy 19(11): 596.

Yuan, B., C. Zhang and X. Shao (2015). "A late acceptance hill-climbing algorithm for balancing two-sided assembly lines with multiple constraints." Journal of Intelligent Manufacturing 26(1): 159-168.

Zhang, Z., K. Wang, L. Zhu and Y. Wang (2017). "A Pareto improved artificial fish swarm algorithm for solving a multi-objective fuzzy disassembly line balancing problem." Expert Systems with Applications 86: 165-176.

Zhu, L., Z. Zhang and Y. Wang (2018). "A Pareto firefly algorithm for multi-objective disassembly line

balancing problems with hazard evaluation." International Journal of Production Research 56(24): 7354-7374.

Özceylan, E., C. B. Kalayci, A. Güngör and S. M. Gupta (2018). "Disassembly line balancing problem: a review of the state of the art and future directions." International Journal of Production Research: 1-23.

Highlights

- Sequence-dependent U-shaped disassembly line balancing problem (SUDLBP) is introduced
- SUDLBP is modelled mathematically and iterated local search (ILS) is developed
- Efficiency of disassembly lines can be improved converting them into a U-line
- A comprehensive study is carried out to test the performance of the proposed models
- ILS outperforms well-known meta-heuristics including SA, TS, GA, PSO and ABC

Graphical Abstract