P-ISSN 2586-9000
E-ISSN 2586-9027
Homepage : https://tci-thaijo.org/index.php/SciTechAsia
Science & Technology Asia
Vol.29 No.1 January - March 2024
Page: [102-111]
Original research article

Two-Sided Disassembly Line Balancing Design with a Soft Computing Technique

Choosak Pornsing[1], Arnut Watanasungsuit[2], Choat Inthawongse[3,*]

[1]*Department of Industrial Engineering and Management, Faculty of Engineering and Industrial Technology, Silpakorn University, Nakhon Pathom 73000, Thailand*
[2]*Hydrocarbon Solutions (Thailand) Company Limited, Thawi Wattana, Bangkok 10170, Thailand*
[3]*Program in Smart Manufacturing Technology, Faculty of Industrial Technology, Muban Chom Bueng Rajabhat University, Ratchaburi 70150, Thailand*

Received 11 October 2023; Received in revised form 11 January 2024
Accepted 1 February 2024; Available online 31 March 2024

ABSTRACT

The aims of this research are twofold. First, a mathematical model is developed for the problem of two-sided balancing of disassembly lines for medium-sized products. Second, a metaheuristic algorithm is proposed for solving such a problem. The proposed mathematical model attempts to minimize the number of workstations, paired stations, modified index of work relatedness and workload balance between workstations. The proposed model falls in the domain of mixed-integer linear programming, which can be explained by the fact that it is a NP-hard problem. We have presented the modified particle swarm optimization, which adds the concept of Pareto optimality to include the non-dominated solutions in the elite list. The proposed method is compared with two competing algorithms: the genetic algorithm and the combinatorial optimization with random algorithm on four benchmark instances. The result, which follows the concept of Pareto optimality, shows that our proposed method provides more non-dominated solutions than the competing algorithms. Moreover, the proposed algorithm exhibits better convergence and diversity than the competing algorithms. In practice, this research work could serve as a basis for designing a two-sided disassembly line for a medium-sized product.

Keywords: Artificial intelligence; Disassembly line balancing; Metaheuristic; Particle swarm optimization; Soft computing

1. Introduction

Due to strict government regulations and customer awareness of the environmental problems of end-of-life products, product recovery is more interesting today than in the past. Technically, product recovery aims to minimize the amount of waste sent to landfills by recovering materials through recycling, dismantling, sorting and refurbishing to bring the product

***Corresponding author:** choatint@mcru.ac.th

to a desired quality level [1]. In addition, disassembly has been shown to play an important role in material and product recovery among all product recovery processes, as it allows selective separation of desired parts and materials [2].

Theoretically, dismantling is a methodical recovery of valuable parts from discarded products through operations [3]. Objectives may include recovering valuable components and sub-assemblies, eliminating hazardous parts, recovering parts from the remainder of the product that can be added to the inventory for later use, achieving environmentally friendly manufacturing standards, recovering parts or sub-assemblies of discarded products to meet a sudden need for these parts, and reducing the amount of residual material going to a landfill. Both industrial and environmental aspects must therefore be taken into account when developing efficient dismantling lines.

The disassembly line balancing problem (DLBP) is a multi-objective problem. As described in [4], the DLBP is a problem of assigning disassembly tasks to workstations in such a way that one or more performance criteria are optimized. It has been mathematically proven to be NP-complete, so the goal of achieving an optimal balance is very computationally intensive [5]. Moreover, the complexity is high when the products that need to be decomposed are extensive, such as compressor systems, motorcycles, ATVs, etc.

Accordingly, in this study we need to model the equilibrium problem of the disassembly line for medium and large products. We consider the character of the disassembly line as a parallel line. Moreover, we propose a metaheuristic method to solve the problem.

2. Literature Review

2.1 Disassembly line considerations

Disassembly is the methodical extraction of valuable parts, sub-assemblies and materials from end-of-life products through a series of steps. Disassembly is the opposite of assembly. Its workflow is different. In a normal assembly environment, the flow process is convergent. The flow process of disassembly, on the other hand, is divergent, as shown in Fig. 1.

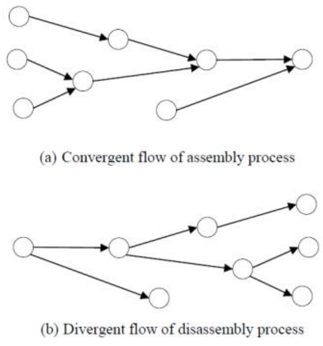
(a) Convergent flow of assembly process
(b) Divergent flow of disassembly process

Fig. 1. Flow process of assembly and disassembly.

The disassembly process is far more laborious than the assembly process. The structure and quality of the returned products are subject to great uncertainty. The condition and reliability of the products are unknown in advance. In addition, some components may cause contamination or be hazardous [6]. The dismantling of cars, for example, contains various hazardous parts: batteries, airbags, headlights, fuel, and engine oil. These obstacles can also lead to complications when balancing the disassembly line.

Reference [7] compares the assembly line (AL) and disassembly line (DL) using 24 indices ranging from tangible indices to intangible indices as shown in Table 1.

Disassembly line balancing (DLB) can be described as the assignment of disassembly tasks to workstations so that all disassembly precedence relationships are satisfied and some effectiveness is optimized [8]. DLB is crucial for minimizing the use of valuable resources (such as time and money) invested in disassembly and for maximizing the degree of automation of the disassembly process and the quality of the recovered parts.

Table 1. Comparison of AL and DL.

Considerations	AL	DL
Demand	Dependent	Dependent
Demand sources	Single	Multiple
Multiple products	Yes	Yes
Demand entry	End product	Individual
Precedence relationships	Yes	Yes
Complexity related to precedence relationships	Low	High
The complexity of performance measures	Moderate	High
Uncertainty related to the quality of components	Low	High
Uncertainty related to the quantity of components	Low	High
Uncertainty related to workstations and material handling system	Low to Moderate	High
Line flexibility	Low to Moderate	High
Layout alternatives	Multiple	Multiple
Forecasting requirements	Single end item	Multi-item
Planning horizon	Product life cycle	Indefinite
Design orientation	Design for assembly	Design for disassembly
Facilities and capacity planning	Straightforward	Intricate
Manufacturing system	Dynamic	Dynamic
Operations complexity	Moderate	High
Flow process	Convergent	Divergent
The direction of material flow	Forward	Reverse
Inventory by-products	None	Potentially numerous
Known techniques for line optimization	Numerous	Some
Know performance measures	Numerous	N/A
Problem complexity	NP-hard	NP-hard
Demand	Dependent	Dependent

Excerpted from [7].

2.2 Mathematical model for DLB

The mathematical model proposed here is a multiobjective optimization problem (MOOP). Since we need to achieve four dimensions of the objective, all measurements are therefore the objectives of this mathematical model [9-10].

Parameters:

CT	= cycle time,
t_i	= processing time of task i,
d_i	= demand of part that is released by task i,
K	= maximum number of workstations that can be used,
$PAND_i$	= index set of AND predecessors of task i,
POR_i	= index set of OR predecessors of task i,
i	= index of tasks $i = 1, 2, ..., M$,
k	= index of workstations $k = 1, 2, ..., K$,
t	= index of periods $t = 1, 2, ..., S$,

Decision variables:

x_{ikt} = 1, if task i is assigned to workstation k in period t; 0, otherwise,

$$\text{Min } z_1 = N_M, \tag{2.1}$$

$$\text{Min } z_2 = N_W, \tag{2.2}$$

$$\text{Min } z_3 = N_W - \frac{N_W}{\sum_{j=1}^{N_W} SN_j}, \tag{2.3}$$

$$\text{Min } z_4 = \frac{N_W}{N_W - 1} \sum_{k=1}^{LL} \sum_{b=L}^{R} \left(\frac{S_{kb}}{WIT} - \frac{1}{N_W} \right)^2, \tag{2.4}$$

subject to

$$\sum_{k=1}^{K} \sum_{t=1}^{S} x_{ikt} \ge d_i, \tag{2.5}$$

$$\sum_{i=1}^{m} t_i x_{ikt} \le CT, \ \ \forall k, t, \tag{2.6}$$

$$\sum_{k=1}^{K} x_{ikt} \le \sum_{k=1}^{K} x_{lkt}, \ \ \forall i, k, t, l \in PAND(i), \tag{2.7}$$

$$x_{ikt} \le \sum_{k=1}^{K} \sum_{b \in POR(i)} x_{bht}, \forall i, k, t, \tag{2.8}$$

$$\sum_{k=1}^{K} x_{ikt} \le 1, \ \ \forall i, t, \tag{2.9}$$

$$x_{ikt} \ge 0 \text{ and integer } \forall i, k, t \tag{2.10}$$

Eqs. (2.1)-(2.4) minimize all relevant objectives: Reduction in the number of paired stations, number of workstations, modified work relatedness index, and workload balance between workstations. Eq. (2.5) is activated when customers demand parts/assemblies; on the other hand, it is relaxed when no minimum is required by customers. The cycle time limit should be observed with Eq. (2.6). Eq. (2.7) indicates that the AND link is fulfilled. These conditions allow the assignment of task i to station *k* if and only if all its AND predecessors are assigned to stations 1 to *k*.

Eq. (2.8) states that the POR precedence relations must be fulfilled. This constraint allows the assignment of task *i* to station *k* if and only if at least one of the *POR* predecessors of task *i* is assigned to stations 1 to *k*. Eq. (2.9) states that a task can be assigned to at most one station in each period. Eq. (2.10) ensures that the decision variables should be binary. Note that $x_{ikt} \le 1$ is enforced by Eq. (2.9).

The mathematical model is a mixed-integer linear programming with a multi-objective function. However, it is inevitable that the problem is NP-hard and does not lend itself to the use of an exact algorithm. We only want to show here that the problem at hand can be modeled by mathematics. So, we use a metaheuristic algorithm to solve such a problem.

3. Research Methodology

3.1 Proposed method

We propose particle swarm optimization to solve the problem at hand. The proposed PSO is based on discrete particle swarm optimization [10].

Particle swarm optimization (PSO), one of the latest metaheuristics that employs a population-based stochastic optimization technique, was originally developed by [11]. PSO is inspired by the metaphor of the social behavior of large flocks of animals, e.g., flocks of birds or schools of fish searching for food. Large populations of birds, for example, typically fly around without a specific destination and then spontaneously form flocks. By maintaining relative distances between individuals, flocking behavior can be synchronized. If you are looking for food in a particular area without knowing the exact location, it is easiest to follow the bird that is known to fly closest to the food. Although PSO is an evolutionary computational method, compared to other evolutionary methods, e.g. GA, it is easier to implement and converges faster because it does not use evolutionary operators such as crossover and mutation. In other words, there is no direct combination manipulation of individuals in the population to generate new offspring in PSO. In addition, PSO is robust and requires fewer parameter settings and less computational memory.

In the jargon of the PSO community, a single potential solution to the problem to be optimized is called a particle (bird), a swarm is called a population, the area to be explored is called the search space of the problem and the food is called the optimal solution. Each particle has its own position (current solution), velocity (trajectory indicating both the size and direction of the flight towards the optimal solution) and a fitness value (measure of performance calculated by a given fitness function and specific to the problem). The particle flies through the multidimensional problem space at a speed that is regularly adjusted by using the navigation cues from its best flight experience (local best) and the best flight experience of the entire population (global best). The former is the so-called cognitive part of the particle, while the latter is the social part.

Let N_p = the number of particles in the population (size of the swarm) and $X_i^t = \left(x_{i1}^t, \dots, x_{iD}^t\right), x_{ij}^t \in \{0,1\}$, be the position of particle *i* in the search space from D dimension at iteration t. The D-dimensional velocity for particle i at iteration t is represented by $V_i^t = \left(v_{i1}^t, \dots, v_{iD}^t\right), v_{ij}^t \in R.$ The position X_i^t represents a solution to the problem, while the velocity V_i^t indicates the rate of change for the position of particle i in the next iteration. Let $P_i^t = \left(p_{i1}^t, \dots, p_{iD}^t\right), p_{ij}^t \in R.$ be the best solution that particle *i* visited $(Lbest)$ and $P_g^t = \left(p_{g1}^t, \dots, p_{gD}^t\right)$ be the best position of the best particle in the swarm at iteration

$(Gbest)$. At iteration t, each particle i adjusts its velocity v_{ij}^{t} based on its previous velocity v_{ij}^{t-1}, its previous best particle position p_{ij}^{t-1}, and the global best particle position p_{gj}^{t-1}, as shown in Eq. (2.11). Then each particle position x_{ij}^{t} is updated according to Eq. (2.12).

$$v_{ij}^{t} = \omega v_{ij}^{t-1} + c_1 r_1 \left(p_{ij}^{t-1} - x_{ij}^{t-1} \right) + c_2 r_2 \left(p_{ij}^{t-1} - x_{ij}^{t-1} \right), \tag{2.11}$$

$$x_{ij}^{t} = x_{ij}^{t-1} + v_{ij}^{t}, \tag{2.12}$$

where c_1 and c_2 are two positive constants that indicate the relative influence of the cognition and social learning factors, respectively. These constants control the maximum step size that the particle can move. r_1 and r_2 are two real random numbers drawn from a uniform distribution $U[0,1]$ generated independently at each iteration, and x is the inertia weight to regulate the effects of previous historical velocities on the current one. The value of x indicates a trade-off between the global and local search capabilities of the swarm. When x is high, the search capability of the particle is more focused on global exploration, while a low x is more conducive to local exploration. In binary discrete particle swarm optimization, the velocity is updated according to Eq. (2.11), but the probability that x_{ij}^{t} takes the value 1 is determined by applying the sigmoid function to the velocity components:

$$s\left(v_{ij}^{t}\right) = \frac{1}{1+e^{-v_{ij}^{t}}}. \tag{2.13}$$

If a random number $r \in 0,1 < s\left(v_{ij}^{t}\right)$, then $x_{ij}^{t} = 1$; otherwise, $x_{ij}^{t} = 0$. It is recommended that the value of v_{ij}^{t} be limited to the range between $[-V_{\max}, +V_{\max}]$ to prevent the sigmoid function from approaching 0 or 1, which would result in the particles leaving the search space. Hence, whenever the updated velocity $v_{ij}^{t} > V_{\max}; v_{ij}^{t}$ is set to Vmax. The value of $V_{\max}$ is normally set to 4. After several iterations, the particles in the swarm tend to move toward better positions, and finally the best position (optimal solution) can be found through their interactions via the communication and corporation of the whole population. The procedure of the proposed PSO is shown in Fig. 2.

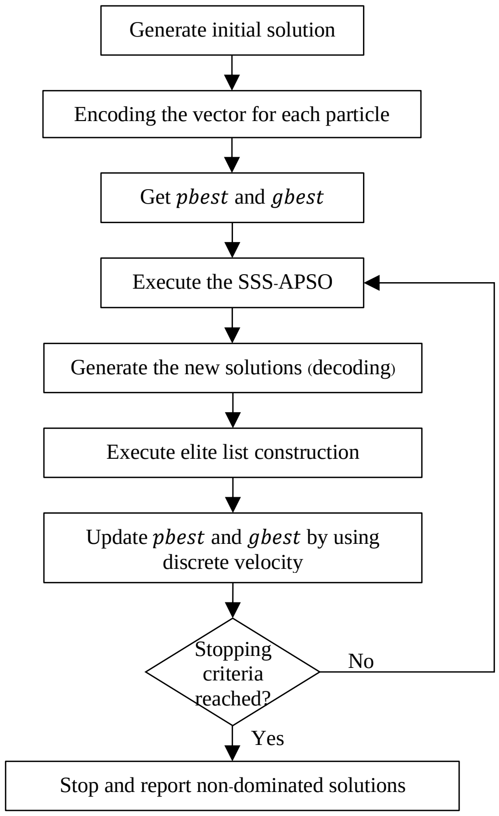
Generate initial solution
Encoding the vector for each particle
Get pbest and gbest
Execute the SSS-APSO
Generate the new solutions (decoding)
Execute elite list construction
Update pbest and gbest by using discrete velocity
Stopping criteria reached?
No
Yes
Stop and report non-dominated solutions

Fig. 2. Modified SSS-APSO for MOOP.

The encoding and decoding methods are crucial procedures for the use of a metaheuristic technique. Suppose Fig. 3 shows the strings of the swarm (two particles in the swarm), the string S_1 is a solution, which is a feasible solution. Decoding is about designing the disassembly line. The tasks are assigned in the order of the string. The position left or right depend on the data L or R of the task. However, the task for attribute E is randomly assigned to the left or right side.

String $S_1 = [2 \quad 3 \quad 5 \quad 1 \quad 7 \quad 10 \quad 8 \quad 9 \quad 4 \quad 6]$

String $S_2 = [5 \quad 1 \quad 4 \quad 7 \quad 10 \quad 2 \quad 8 \quad 9 \quad 3 \quad 6]$

Fig. 3. Two-solution strings.

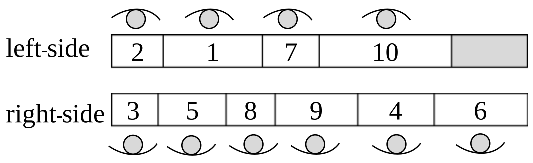

Fig. 4. Disassembly line of string S_1.

The mechanism used in this study to fulfill all constraints is the penalty method, where a penalty is added to the objective function to punish an individual for violating a constraint. We chose a large number *M* to penalize.

3.2 Benchmark problems

The benchmark problems were taken from literature. The ten-part product comes from [12] and the cell phone was provided by [13]. These two cases are small products that do not need to be processed in a two-sided disassembly line. However, we need to test the performance of the proposed method in general cases and check how it works for simple cases.

The case of the car engine is a medium-sized product that must be processed on a two-sided disassembly line. This case was taken from [14]. The washing machine was provided by this study. It is a top-loading washing machine with dryer, model NA-W105T of the Panasonic brand, which consists of 39 parts/components/assemblies, none of which represent hazardous contents. Fig. 3 shows the washing machine prepared in this study. Table 3 shows the detailed disassembly process.

Table 2. Benchmark instances.

Product	Total task time (units)	Cycle time (units)	Number of operations	Product size
Ten-part	170	13	10	small
Cell-phone	142	10	25	small
Automobile engine	3,636	200	35	medium-large
Washing machine	936.93	50	39	medium-large

Fig. 3. Top-loaded washing machine.

Table 3. Washing machine instance.

Task	Part	Time	Direction	Precedence part
1	Bolts (2)	5.72	L	-
2	Bolt (1)	6.74	R	-
3	Cover	2.75	L	-
4	Panel	3.36	E	1, 2
5	Wash timer wiring	25.22	L	4
6	Spin timer wiring	38.64	R	4
7	Washing timer knob	3.43	L	5
8	Spin timer knob	3.03	R	6
9	Timer switch bolts (2)	35.61	R	4
10	Bind Tapping (5)	93.8	L	4
11	Bind Tapping (5)	177.05	R	4
12	Washing timer switch	4.59	L	10, 11
13	Spin timer switch	3.19	R	9
14	Washing selector knob	4.56	L	13
15	Cycle selector knob	3.46	R	12, 14
16	Buzzer	4.56	R	10, 11
17	Pannel A	4.16	E	12, 13
18	Switch cover	3.73	L	17
19	Spinner lid	3.17	R	17
20	Body b plate bolts (2)	10.72	R	19
21	Nozzle holder bolts (2)	3.19	L	18
22	Body b (3)	4.90	L	3, 19

Task	Part	Time	Direction	Precedence part
23	Nozzle holder	4.55	L	21
24	Body b plate	3.56	R	20
25	Body b	23.63	R	22
26	Over flow filter a	2.95	L	25
27	Special bolt	17.51	L	3
28	Pulsator unit	2.52	L	27
29	Back panel bolt (1)	2.53	R	-
30	Back panel	10.58	R	29
31	Back panel bolts (2)	9.14	L	-
32	Back panel	2.73	L	31
33	Base a bolts (3)	2.43	R	-
34	Bolt	301.25	R	30
35	Tub a	30.77	E	30, 33
36	Drain tube	7.45	L	30
37	Motor bolts (3)	3.7	L	32, 35
38	Electric wire	62.39	L	30
39	Motor	3.69	L	35, 37

3.2 Experimental design

In this study, we selected the genetic algorithm (GA) [15] and the coincidence algorithm (COIN) [16] as competing algorithms because they have been used to solve problems such as the present one. Many researchers have also used these algorithms to solve combinatorial optimization problems. However, the mechanisms of the encoding and decoding methods have been modified to be consistent with our proposed method to allow a fair comparison.

All procedures were coded in Python 3.8, 64-bit on Anaconda 3, Jupyter IDE, and run on an Intel Core i7-8750H CPU@2.20 GHz, 8.00 GB RAM personal computer under MS. Windows 10 Pro. However, we did not intend to measure the computing time. The parameter settings are listed in Table 4.

Four measured values were determined: number of paired stations, number of jobs, modified labor relations index, and workload balance between jobs. We then used Pareto optimality theory to find the Pareto-optimal front containing the solutions in the elite list. Note that each combination of instance and solution algorithm was run 30 times to eliminate random deviations.

Table 4. Parameter settings.

Parameter settings	Proposed PSO	GA for DLB	COIN for DLB
Population size	100	100	100
Crossover method	-	One-point	-
Mutation method	-	Reciprocal exchange	-
Crossover probability	-	0.7	-
Mutation probability	-	0.2	-
c_1 , c_2	0.15, 0.20	-	-
ω	0.9	-	-
Iterations	1000	1000	1000

4. Result and Discussion

The exciting achievement in this study is the way the algorithm has so far found the non-dominated solution. From a quantitative point of view, we were interested in the number of non-dominant solutions, although the quality of the solution was measured by the inverted generation distance (IGD). The IGD was invented to measure the convergence and diversity of evolutionary algorithms with multiple and many objectives [17].

Let S be a result solution set of a multi-objective evolution algorithm on a given multi-objective optimization problem. Let R be a set of uniformly distributed representative points of the Pareto front. The IGD value of S relative to R can be calculated as [18].

$$IGD = (S,R) = \frac{\sum_{r\in R} d(r,S)}{|R|}, \tag{4.1}$$

where $d(r,S)$ is the minimum Euclidean distance between r and the points in S, and $|R|$ is the cardinality of R. Note that the points in R should be well distributed and $|R|$ should be large enough to ensure that the points in R can represent the Pareto front. This guarantees that the IGD value of S is able to measure the convergence and diversity of the solution set. The lower the IGD value of S is, the better its quality is.

Table 5 shows the numerical results of the non- dominant solution found so far. Ideally, many Pareto front solutions show the powerful search method — maximum, minimum, average and standard deviation of the Pareto front solutions of 30 runs.

Table 5. Numerical experiment results.

Instances	Pareto front solutions	Proposed PSO	GA for DLB	COIN for DLB
Ten-part	max	13	15	17
	min	6	5	6
	$\bar{x}$	8.90	11.87	**12.27**
	σ	2.50	2.37	3.61
Cell-phone	max	10	12	12
	min	3	3	4
	$\bar{x}$	6.67	**9.03**	5.57
	σ	2.17	2.89	2.20
Automobile engine	max	8	5	3
	min	3	3	1
	$\bar{x}$	**5.73**	4.03	1.80
	σ	1.36	0.81	0.66
Washing machine	max	9	6	7
	min	5	1	3
	$\bar{x}$	**7.07**	2.97	4.10
	σ	1.58	1.43	0.99

Table 5 shows that the COIN method works well for small products. It does not require a bipartite decomposition line, so the proposed decoding technique relies on an ordinary one. COIN also provided satisfactory results. GA was competitive on the cell phone product. However, COIN showed better performance as it has minimum standard deviation. The proposed method outperformed the competing methods on large products. It provided an exceptional number of non-dominated solutions while maintaining low deviation. Fig. 4 is an example of the Pareto front of the proposed method. Moreover, low standard deviation in evolutionary algorithms means that the search process tends to be trapped in a local minimum.

As already mentioned, we also examined the quality of the solutions. Table 6 shows the IGD results from 30 runs. Interestingly, all combinations in the computational experiments showed that the proposed method yielded exceptional solutions by providing fewer values for the inverted generation distance.

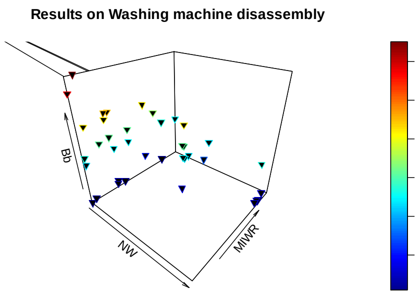

Fig. 4. Pareto-optimal frontier of washing machine disassembly instance.

Table 6. Inverted generational deviation summary.

Instances	IGD	Proposed PSO	GA for DLB	COIN for DLB
Ten-part	$\bar{x}$	**4.0312E+00**	1.3274E+01	1.5684E+01
Cell-phone	$\bar{x}$	**3.9452E+00**	9.8320E+00	1.0752E+01
Automobile engine	$\bar{x}$	**1.5601E+01**	8.3249E+01	1.0532E+02
Washing machine	$\bar{x}$	**1.8329E+0.1**	9.8650E+01	8.7886E+01

We want to analyze the results using a statistical technique. The analysis of variance (ANOVA) was used to quantify the significant differences between the results of the three approaches. Tables 7 to 8 show the results of the analysis of variance from the data analysis package in the Excel program. At a significant level of $\alpha = 0.05$, the results show that all approaches provide significantly different results. GA and COIN work well for small products, while our proposed technique outperforms the competing methods for medium to large products.

Table 7. Analysis of variance table for ten-part instance.

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*

Between Groups	202.9556	2	101.4778	11.9299	2.64E-05	3.1012
Within Groups	740.0333	87	8.50613			
Total	942.9889	89				

Table 8. Analysis of variance table for cell phone instance.

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	84.9555	2	42.4777	7.0986	1.39E-03	3.1012
Within Groups	520.6	87	5.9839			
Total	605.5556	89				

Table 9. Analysis of variance table for automobile engine instance.

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	215.2667	2	107.6333	109.9933	1.51E-24	3.1012
Within Groups	85.1333	87	0.9785			
Total	300.4021	89				

Table 10. Analysis of variance table for washing machine instance.

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	259.6222	2	129.8111	70.73215	5.76E-19	3.101296
Within Groups	159.6667	87	1.835249			
Total	419.2889	89				

5. Conclusion

The design of a two-sided disassembly line is difficult to solve. Moreover, multi-objective optimization was considered in this study since there were many objectives to optimize. The proposed mathematical multi-objective model was proposed with a given cycle time. Then, the number of paired stations, work centers, modified work relation index, and workload distribution between work centers were minimized.

A modified particle swarm optimization (SSS-APSO) was proposed, and a discretization mechanism was introduced. It constructs an elite list by storing the non-dominated solutions found so far to form the Pareto front. Four benchmark products and two competing algorithms were used. The results of the computational experiments showed that the proposed method outperformed the competing algorithm in terms of quantity and quality for medium to large products. However, for small products, it was not better than the two competing algorithms. The advantages of the proposed algorithm are that it is easy to code and suitable for multiobjective optimization problems. However, the disadvantage is that it tends to get stuck in a local optimum for small problems since it has low performance in the exploitation phase. In future research, we need to modify the proposed algorithm to perform well in both exploration and exploitation phases in a search course. Nevertheless, the proposed approach could facilitate a practitioner to plan a dismantling plant for medium to large-sized products in a natural industrial environment. Nevertheless, the proposed approach could make it easier for a practitioner to plan a dismantling plant for medium to large-sized products in a natural industrial environment.

Acknowledgments

The authors thank the editor and the anonymous reviewers for their careful review of our manuscript and for many insightful comments and suggestions in the earlier version. This research project was funded by the Silpakorn Universtiy Research, Innovation and Creative Fund.

References

[1] Özceylan E, Kalayci CB, Güngör A, Gupta SM. Disassembly line balancing problem: a review of the state of the art and future directions. International Journal

of Production Research. 2019 Aug 29;57(15-16):4805-27.

[2] Hezer S, Kara Y. A network-based shortest route model for parallel disassembly line balancing problem. International Journal of Production Research. 2015 Mar 19;53(6):1849-65.

[3] Kalayci CB, Gupta SM. A particle swarm optimization algorithm with neighborhood-based mutation for sequence-dependent disassembly line balancing problem. The International Journal of Advanced Manufacturing Technology. 2013 Oct;69(1):197-209.

[4] Kucukkoc I. Balancing of two-sided disassembly lines: Problem definition, MILP model and genetic algorithm approach. Computers & Operations Research. 2020 Dec 1;124:105064.

[5] Kalayci CB, Hancilar A, Gungor A, Gupta SM. Multi-objective fuzzy disassembly line balancing using a hybrid discrete artificial bee colony algorithm. Journal of Manufacturing Systems. 2015 Oct 1;37:672-82.

[6] McGovern SM, Gupta SM. A balancing method and genetic algorithm for disassembly line balancing. European journal of operational research. 2007 Jun 16;179(3):692-708.

[7] Hezer S, Kara Y. A network-based shortest route model for parallel disassembly line balancing problem. International Journal of Production Research. 2015 Mar 19;53(6):1849-65.

[8] Ding LP, Feng YX, Tan JR, Gao YC. A new multi-objective ant colony algorithm for solving the disassembly line balancing problem. The International Journal of Advanced Manufacturing Technology. 2010 May;48(5):761-71.

[9] McGovern SM, Gupta SM. Disassembly Line: Balancing and Modeling. McGraw-Hill Education; 2011.

[10] Pornsing C, Watanasungsuit A. Discrete particle swarm optimization for disassembly sequence planning. In2014 IEEE International Conference on Management of Innovation and Technology IEEE. 2014, pp. 480-5.

[11] Kennedy J, Eberhart R. Particle swarm optimization. In Proceedings of ICNN'95-international conference on neural networks IEEE. 1995;4:1942-8.

[12] Kongar E, Gupta SM. Genetic algorithm for disassembly process planning. In Environmentally Conscious Manufacturing II SPIE. 2002;4569:4-62.

[13] Gupta SM, Erbis E, McGovern SM. Disassembly sequencing problem: a case study of a cell phone. In Environmentally conscious manufacturing IV SPIE. 2004: 5583:43-52.

[14] Seidi M, Saghari S. The balancing of disassembly line of automobile engine using genetic algorithm (GA) in fuzzy environment. Industrial Engineering and Management Systems. 2016;15(4):364-73.

[15] Gulivindala AK, Bahubalendruni MV, Chandrasekar R, Ahmed E, Abidi MH, Al-Ahmari A. Automated disassembly sequence prediction for industry 4.0 using enhanced genetic algorithm. Computers, Materials & Continua. 2021 Jan 1;69(2):2531-48.

[16] Wattanapornprom W, Olanviwitchai P, Chutima P, Chongstitvatana P. Multi-objective combinatorial optimisation with coincidence algorithm. In2009 IEEE Congress on Evolutionary Computation IEEE. 2009, pp. 1675-82.

[17] Y. Sun, G. G. Yen and Z. Yi, "IGD Indicator-Based Evolutionary Algorithm for Many-Objective Optimization Problems," in IEEE Transactions on Evolutionary Computation 2019; 23(2):173-87.

[18] Zhang Y, Zeng B, Li Y, Li J. A mullti-or many-objective evolutionary algorithm with global loop update. arXiv preprint arXiv:1803.06282. 2018 Jan 25.