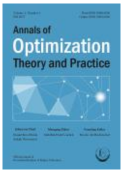
Annals of Optimization Theory and Practice

Volume 3, Number 4, 65-82
December 2020

DOI: 10.22121/aotp.2021.264516.1059



Optimizing the first type of U-shaped assembly line balancing problems

Pouria Pourmomen Davani[1] • Ahmad Wael Mahmoud Kloub[2] • Mazyar Ghadiri Nejad[2]✉

[1] Computer Engineering Department, Eastern Mediterranean University, Via Mersin 10, Famagusta, TRNC, Turkey

[2] Industrial Engineering Department, Girne American University, Via Mersin 10, Kyrenia, TRNC, Turkey

✉ mazyarghadirinejad@gau.edu.tr

(Received: December 1, 2020 / Accepted: December 26, 2020)

Abstract In the literature, there are various types of assembly line balancing problems. Consequently, different types of solution approaches such as exact, heuristic, and metaheuristics have been proposed to solve such problems. In this research, we are going to propose a metaheuristic solution method based on applied grouping evolution strategies to solve the u-shaped assembly line balancing problem where the aim is the minimization of the number of workstations considering a given cycle time for the assembly line. By introducing the just-in-time (JIT) production principle, it can be proven that the U-shaped assembly line system has a better performance than the traditional straight-line system. Different test problems from the literature are solved and key indexes like the line efficiency, smoothness index, and variation, are calculated for the problems. Then the proposed method is compared to one of the recent solutions approached based on the genetic algorithm. The results show that the proposed method has the potential t be considered as one of the most efficient methods in this field.

Keywords Assembly line balancing; U-shaped layout; Ranked positional weight method; COMSOAL; Grouping evolution strategy algorithm

1. Introduction

Assembly line is defined as the arrangement of workstations where different parts get together to make a specific final product. The problem of changing the arrangement of workstations in a way that the optimum throughput and/or performance, upon some specific criteria, are gained is called an assembly line balancing problem (ALBP) (Kumar and Mahto, 2013). Usually, the objective of ALBPs is to reduce the number of workstations as much as possible for a given cycle time (Type-I) or to reduce the cycle time given a cycle time (Type-II). ALBPs are classified into simple ALBPs (SALBPs), and General ALBPs (GALBPs) (Becker and Scholl, 2006). SALBPs are the most

Published by Firouzabad Institute of Higher Education, Firouzabad, Fars, Iran

eminent assembly line, which is a serial arrangement of workstations considering that the same product is running on a straight-line layout (Boysen *et al.*, 2007). According to Scholl and Becker (2006), this type of problems are classified into four groups concerning their objectives functions:

- SALBP Type-1 (SALBP-1), which aims to minimize the number of workstations on the line for a fixed cycle time.
- SALBP Type-2 (SALBP-2), which aims to minimize the cycle time for a fixed number of workstations on the line.
- SALBP Type-E (SALBP-E) aims to maximize the efficiency of the line simultaneously minimizing the number of workstations and the cycle time.
- SALBP Type-F (SALBP-F) aims to determine a feasible line for a combination of the number of workstations and cycle time.

Other problems that are not included in SALBPs are considered as GALBPs. Mixed-model assembly line balancing (MALBP) or mixed-model sequencing problem (MSP), and also U-shaped ALBP (UALBP) are categorized as GALBP (Boysen et al., 2007). In various cases, the traditional SALBs have shown inefficiency in line flexibility, job monotony, and large inventories. With the invention of just-in-time (JIT), UALBs have become more popular because of having higher efficiency and more flexibility (Monden, 2011). The U-shaped layout lets the operator work on both front sides of the line, in a workstation, called a crossover workstation. The more crossover workstations, the more flexibility in the ALB will be (Hwang *et al.*, 2008). Additionally, there are better sights of the production line, increasing the personnel oral communications, and the ability of rebalancing in fast-changing of demands or operating environment. The other advantages are productivity improvement, reduction in work-in-process inventory, space requirement, and lead-time are the other benefits of UALBs (Glonegger and Reinhart, 2015). Therefore, SALBs are being replaced by UALBs to adopt industries with JIT philosophy. In the UALBP, a task can be assigned to a station after all of its predecessors or successors have been assigned to stations.

UALBPs are a relatively new and promising topic in the ALBPs literature. One of the first deep studies is related to Miltenburg and Wijngaard (Miltenburg and Wijngaard, 1994) who proposed a dynamic programming formulation to solve 21 relatively small problems, and develop a heuristic procedure based on the maximum ranked positional weight (RPW) for large size problems. Later on, Miltenburg and Sparling (Miltenburg and Sparling, 1995) developed three exact algorithms for the UALBPs: a reaching dynamic programming algorithm, breadth- and depth-first branch-and-bound algorithms. To handle larger problems, Scholl and Klein (1999) proposed the ULINO method based on the branch-and-bound algorithm to solve different versions of ALBPs (Scholl and Klein, 1999). Erel et al. (2001) developed an SA-based algorithm based on an intelligent mechanism to search the large solution space effectively Erel et al., (2001). Furthermore, Gokcen *et al.* (2005) proposed a shortest route formulation (Gökçen *et al.*, 2005), Gokcen and Agpak (2006) and Toklu and Ozcan (2008), developed Goal Programming formulations (Gökçen *et al*. (2006), Toklu and özcan (2008)), and Jayaswal and Agarwal

Published by Firouzabad Institute of Higher Education, Firouzabad, Fars, Iran

(2014) used resource-dependent task times to solve UALBPs (Jayaswal and Agarwal, 2014).

According to the definition proposed by Falkenauer (1994), grouping problems aim to partition the members of a set into different groups where each object is exactly in one group, where the ordering of groups is not relevant. Since all the possible assignments are not acceptable in grouping problems (GPs), hence, GPs are featured by an objective function considering the various combination of groups. Moreover, by using evolutionary algorithms a group segment is the fundamental block that should be considered for the search. Based on this fact, scientists have used evolutionary algorithms to improve GPs (Kashan *et al.*, 2013). GES is a kind of evolutionary algorithm that was recently proposed for crisp GPs. This method is compatible with the Evolution Strategy (ES), proposed by Rechenberg (1978), which uses Gaussian mutation during optimization, while GES uses another similar mutation (Beyer *et al.*, 2002). Some well-known grouping problems are ALBPs (Nejad and Kashan, (2019), Ghadirinejad *et al.*, (2013)), graph coloring (Yuan *et al.*, (2017), Mosa *et al.*, (2017)), bin packing (Pereira, (2016), Li and Zhang, (2018)), flexible robotic cells (Nejad *et al.*, (2018), Nejad *et al.*, (2019), Nejad *et al.*, (2019), Ghadirinejad and Mosallaeipour,(2013)), identical/non-identical parallel-machines scheduling (Mosallaeipour (2018), Mosallaeipour (2018)), delivery and emergency problems (Shavarani *et al.*, (2018), Ghadiri Nejad and Banar, (2018)).

By doing adjusting modification in grouping evolution strategies (GES), proposed by Kashan et al. for NP-hard types of problems, and modifying the proposed method to deal with the ALBPs by Nejad *et al.* (2018), are solved. The modified strategy is applicable for deterministic UALBPs, aiming to optimize the system with the smallest sets of workstations. In this study, regards to study of Hwang *et al.* (2008) and Erel *et al.* (2001), we try to use a modified version of the GES proposed by Kashan *et al.* (2009), and later on by Nejad *et al.* (2018) to solve some UALBP standard test problems from the literature. To find out an enhanced result, we use revised-RPW. The method proposed by Fathi *et al.* (2011).

This study is continued by the following sections: In Chapter 2, the proposed algorithm and all the utilized techniques to improve the proposed method are explained. Chapter 3, contains the computational results and the related discussions, finally the study is ended by concluding the study and proposing some potential future topics.

2. Proposed Algorithm

To achieve the goals of this study, a hybrid algorithm including an exact algorithm to find an initial solution and a grouping meta-heuristic algorithm to improve the solution are developed. Then, the proposed algorithm is coded and solved by MATLAB software, and then by using the standard problems considered in Hwang *et al.* (2008), the quality of the proposed method is measured.

The simple case of a UALBP is one of the most discussed issues in combinational optimization. In this problem precedence graph of activities are given where activity *j* has a processing time of T_j unit. The objective is assigning the activities to workstations considering prerequisite activities with fixed cycle time, aiming to minimize the number of workstations. The proposed algorithm in this research is a two-stage algorithm such as (1) creating an initial solution, and (2) improving the initial solution to achieve the final solution.

2.1. Generating an Initial Solution

There are several methods for determining the initial solution of the UALBP that each of them has its strengths and weaknesses. These methods are included in all exact and heuristic methods that any of them can be considered as the initial solution algorithm. The COMSOAL method proposed by Arcus (1965), as one of the well-known methods in this field, assigns the tasks from the first one in the precedence graph to the workstations randomly, considering the given cycle time. As it is evident at the first glance, one of the strengths of this method besides the performance simplicity is the ability to find different results due to the use of a random selection process for the allocation of the activities in each step. Because of having much flexibility and high-performance power, this method gives the desired result in every. Hence, it is necessary to check the results of several runs to reach the best outcome to calculate the line efficiency and smoothness index.

In contrast, there is a measure to evaluate a new meta-heuristic algorithm to find its performance. Hence, generating the same initial solutions for a problem with constant parameters such as cycle time, processing times, and precedence graph, seems necessary. Therefore, the COMSOAL method has been used only in the second state for improving the initial solution. Leaving aside the COMSOAL method to find a way to create an initial solution that every time gives a constant solution for the same problem, the Ranked Positional Weight (RPW) method Helgeson *et al.* (1961) is considered. After some necessary changes to improve the result, the Revised-RPW method was proposed.

2.1.1 Algorithm of the RPW for UALBP

Based on the steps of the RPW method Helgeson *et al.* (1961), the weight of each task must be calculated in both the forward direction and backward direction. The parameters used in this method are the following:

$T(S_i)$	Total time of each station
$T(x)$	Time of each task
CT	Given cycle time
N	Number of tasks
M	Number of workstations
S	Minimum feasible number of workstation
MCT	Minimum feasible cycle time
$CT*$	Modified cycle time

To apply a priority for each task, we have used a precedence network calculating task’s weights that can be explained as the total of activity time, and times of the various succeed or progress, correspondingly. There are two criteria for assigning tasks to workstations such as succession and precedence priorities and having free space to handle the assigning task to the workstations. In the case of multiple available tasks, the one with the highest weight is assigned. Whenever all tasks have been bounded we say the assignment is complete.

In this solution, *CT* is replaced by a new symbol *CT** which is computed as below:

$$S = \sum_{i=1}^{n} T(x) \,/\, CT \tag{1}$$

If *S* becomes a non-integer value it should be rounded up.

$$MCT = \sum_{i=1}^{n} T(x)/S \tag{2}$$

$$CT^{*} = [(MCT + S)/2] \tag{3}$$

It should be kept in mind that *MCT* < *CT** < *CT*. As it can be concluded *CT* can be chosen freely in the domain *(MCT, CT).* However, choosing *CT** as *CT* yields more appropriate outputs. To maintain the preferred circumstances, the following relations must be satisfied:

$$T(S_i) = \sum_{x \in S_i} T(x) \leq CT \qquad i = 1, \dots, M \tag{4}$$

$$if\ (x, y) \in P, x \in\ S_i\ and\ y \in\ S_j\ then\ i \leq j\ for\ all\ x. \tag{5}$$

$$if\ (y, z) \in P, y \in\ S_j\ and\ z \in\ S_k\ then\ k \leq j\ for\ all\ z. \tag{6}$$

Equation (4) implies that the total task time belongs to a workstation cannot be more than *CT*. Equation (5) means that if task *x* is the precedence of task *y*, its workstation number must not be more than the workstation number of task *y*. Similarly, equation (6) guarantees that there is a possibility to assign a task to a workstation after the workstation that its predecessor belongs to, because of the U-shaped layout configurations. In the following, the steps of the RPW algorithm are explained.

- Step 1: Computing the least quantity of workstations *S*, the least possible cycle time *MCT*, and *CT** = [(*MCT* + *CT*)/2].
- Step 2: Adopting a new workstation and computing every work element's weight in two ways of forwarding and backward direction. After that, the activities, appropriate for assigning, are identified and a candidate list is generated.
- Step 3: Arrange the weight of work elements in descending order.
- Step 4: Assign the first activity containing the highest weight, to the first station.
- Step 5: Calculate the idle time (*IT*) for station *r* that has *k* task with the below formula:

$$IT = CT^* - \sum_{i=1}^{k} t_{\mathrm{ir}} \tag{7}$$

- Step 6: Comparing the time of the first none assigned activity that has the highest positional weight with the *IT* of the last work station, Afterwards, assigning the activity if its process time is not more than the last work station’s *IT*.
- Step 7: Assign an activity to a new workstation if its duration is bigger than the *IT* of the workstation, and go to step 5.

2.1.2 Revised-RPW Method

The RPW method proposed by Helgeson *et al.* (1961), is modified according to the flowchart shown in Figure :

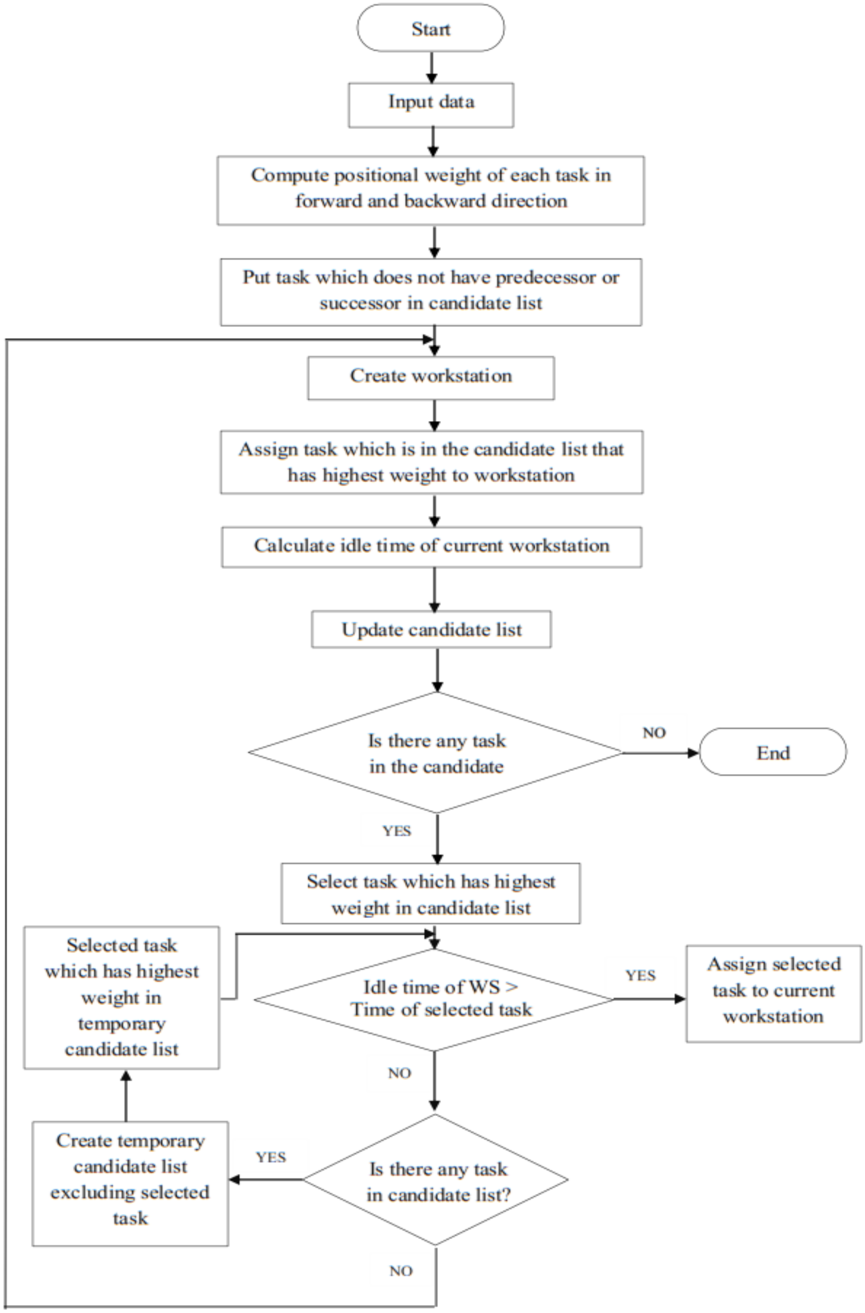

Start
Input data
Compute positional weight of each task in forward and backward direction
Put task which does not have predecessor or successor in candidate list
Create workstation
Assign task which is in the candidate list that has highest weight to workstation
Calculate idle time of current workstation
Update candidate list
Is there any task in the candidate
NO
End
YES
Select task which has highest weight in candidate list
Selected task which has highest weight in temporary candidate list
Idle time of WS > Time of selected task
YES
Assign selected task to current workstation
NO
Create temporary candidate list excluding selected task
YES
Is there any task in candidate list?
NO

Figure 1. Flowchart of Revised-RPW

2.2. Initial Solution Improvement until Achieving Final Solution

To find the optimal solution, the following steps are performed.

2.2.1. Using a meta-heuristic algorithm for finding some alternative solutions

For solving ALBPs, according to NP-hardness of such problems, each of the meta-heuristic algorithm like Genetic Algorithm (Nejad *et al.*, 2018), Tabu Search (Toklu and özcan, 2008), Ant Colony (López-Ibáñez *et al.*, (2015), Dorigo *et al.*, (2006)), Simulated Annealing (Ghadiri Nejad *et al.*, 2018), Local Search (Vizvari *et al.*, 2018), Grey Wolf (Vatankhah Barenji *et al.*, (2018), Nadimi-Shahraki *et al.*, (2021)), Memetic algorithm (Pereira *et al.*, (2018)), etc. may be used. In this study, first, we utilize the GES algorithm to find some alternative solutions. This method applies mutation operator to avoid static solutions, in the way that first, several tasks of the initial solution are removed, then by using a heuristic technique, the missing tasks are assigned and the solution is completed. In Figure 2, we briefly describe all the steps.

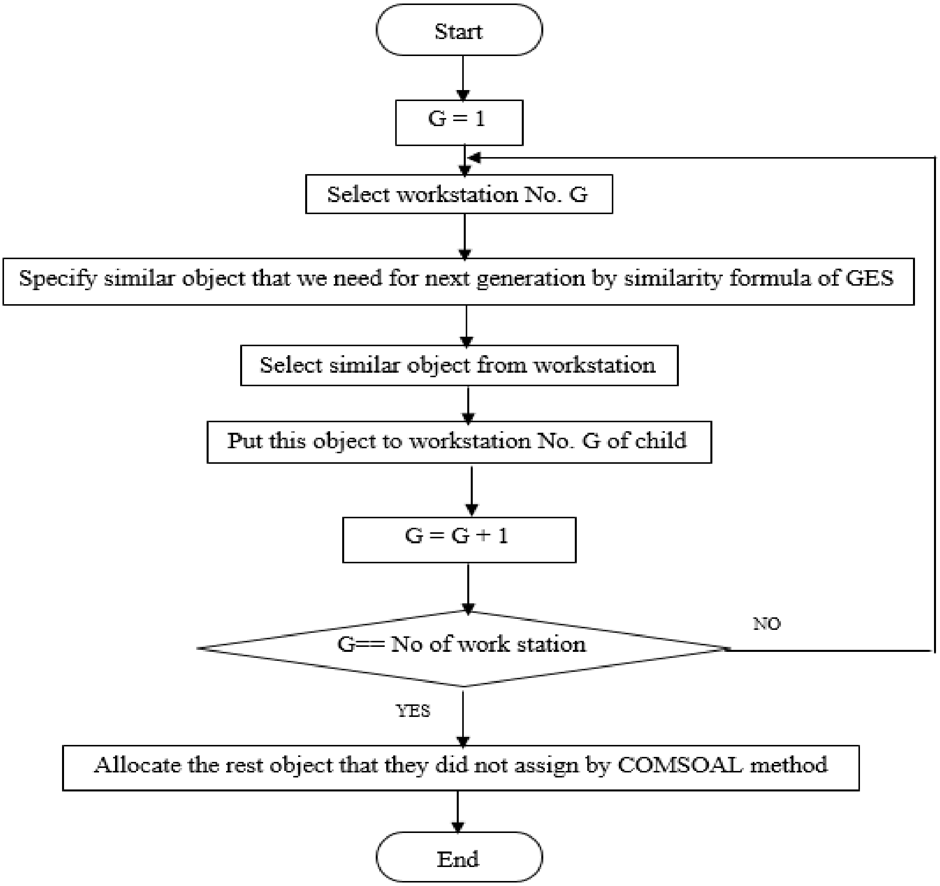

Start
G = 1
Select workstation No. G
Specify similar object that we need for next generation by similarity formula of GES
Select similar object from workstation
Put this object to workstation No. G of child
G = G + 1
G== No of work station
NO
YES
Allocate the rest object that they did not assign by COMSOAL method
End

Figure 2. Flowchart of Mutation Operator

Published by Firouzabad Institute of Higher Education, Firouzabad, Fars, Iran

2.2.2. Using a heuristic method for assigning the activities after mutation

The mutation operator is frequently used in meta-heuristic algorithms, especially in GES to find a better solution without any static result in search. In this method, after removing some assigned activities from related workstations according to a special pattern, the removed tasks are assigned gain to the workstations by a heuristic method. Noted that in this step we could not use revised-RPW because of its unique solution in similar cases. Considering the simplicity and flexibility of the COMSOAL method (Arcus, 1965), we utilized it in this step of the algorithm. The flexibility of this method is because of generating different solutions by a random selection process among activities that are ready to assign in each step.

The classical COMSOAL method starts from the first node (activity) and continues with the nodes that do not have any predecessor. It assigns the activities to workstations considering the idle time of stations. However, in our case, after performing the mutation operator, maybe in the middle of the precedence diagram must be assigned. Therefore, at first, the algorithm must distinguish which activities have no predecessor or successor and after that, it must ignore the assigning of those activities that have previously been assigned. To create these changes in the revised algorithm, there is a constraint for predecessor and successor activities, which their algorithm are as below:

- Step 1: Choose the activities without predecessor or successor that are ready to assign.
- Step 2: Finding the possible workstations which activity can be assigned.
- Step 3: Find the last workstation that contains predecessor activities (MPWS) and successor activities (MSWS) of the chosen task. Then, compare them and choose the minimum one.
- Step 4: Control the assumption that the chosen activity’s workstation number must be equal or bigger than the workstation number that we described in step3.
- Step 5: Remove the assignment of the chosen activity, if the assumption of step 4 has been violated.

Figure 3 illustrates the mentioned steps in a flowchart format. In this flowchart, we considered APWS as an appropriate workstation for assigning activities that are ready to be assigned.

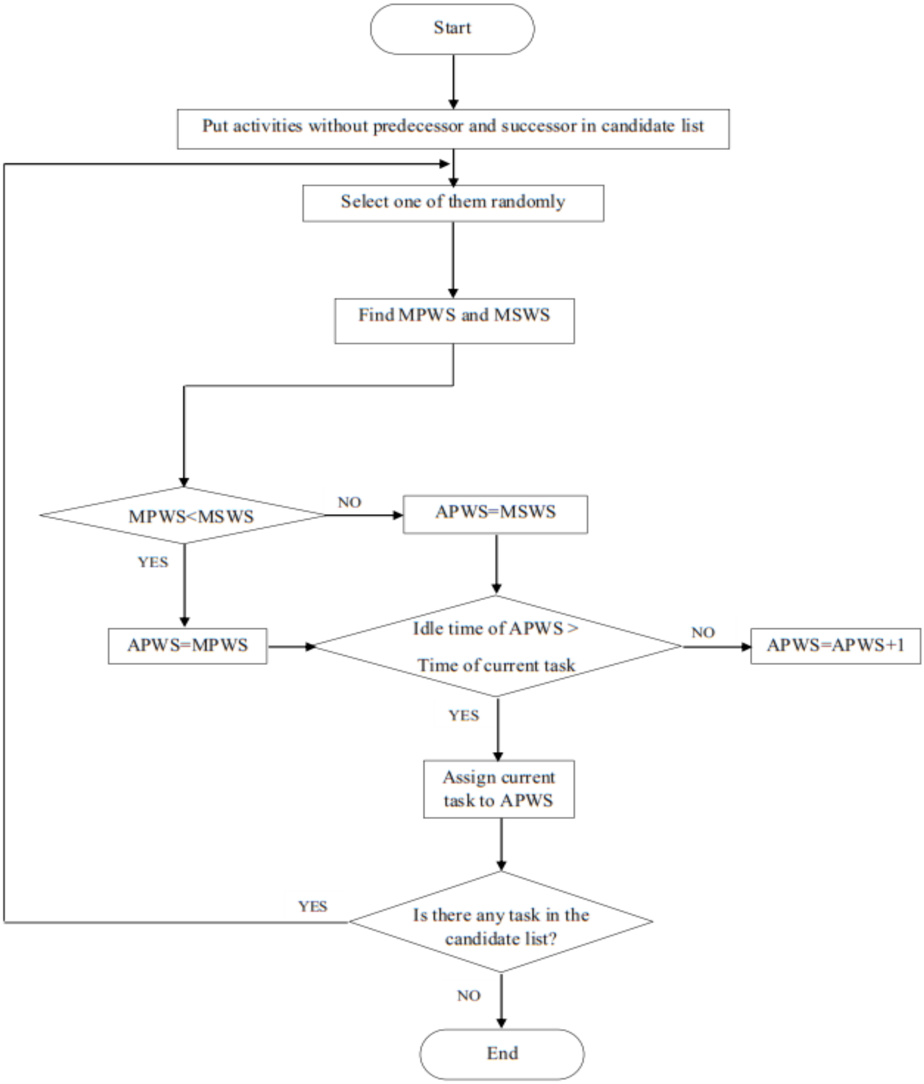

Start
Put activities without predecessor and successor in candidate list
Select one of them randomly
Find MPWS and MSWS
MPWS<MSWS
NO
YES
APWS=MSWS
APWS=MPWS
Idle time of APWS > Time of current task
NO
APWS=APWS+1
YES
Assign current task to APWS
Is there any task in the candidate list?
YES
NO
End

Figure 2.2.2. Flowchart of Revised-COMSOAL

2.3. Using a Method to Select the Best Solution in Every Step

After changing the solution method in the proposed algorithm and improving it in every step, in addition to the chosen result of the previous step, two new solutions are generated. To find the best result for producing the next solution, it is necessary to compare all three solutions in every step. In this regard, a method based on the smoothness index (*SI*), which is explained later, is considered as the selection operator. Figure 4 describes the improvement of the initial solution.

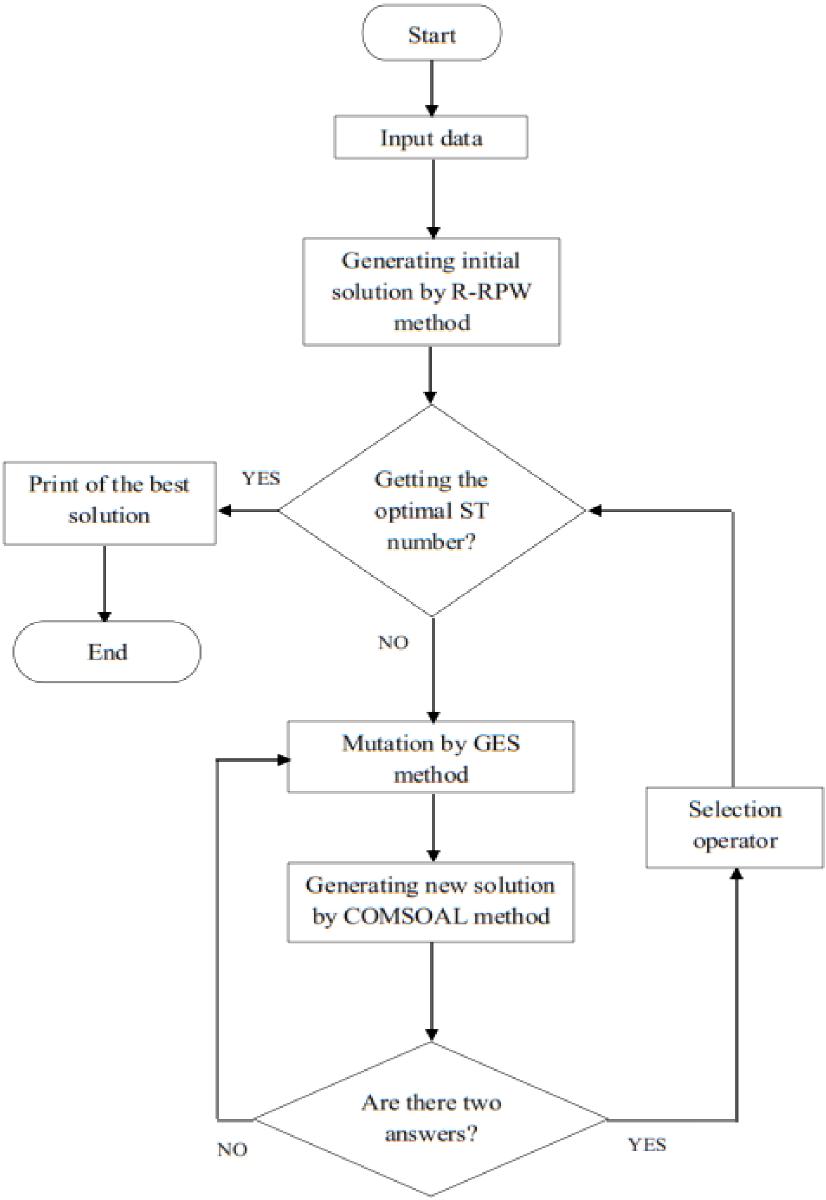

Start
Input data
Generating initial solution by R-RPW method
Getting the optimal ST number?
YES
Print of the best solution
End
NO
Mutation by GES method
Generating new solution by COMSOAL method
Are there two answers?
NO
YES
Selection operator

Figure 4. Flowchart of the proposed algorithm

3. Computational Results

3.1. Simulation Setup and Performance Metrics

In this chapter, the results of the proposed methods are compared with some well-known problems and the solutions for the considered test problems are compared with the best solutions that are already obtained by Hwang *et al.* (2008). The needed data to examine

Published by Firouzabad Institute of Higher Education, Firouzabad, Fars, Iran

the mentioned methods and to compare the results have been acquired from their study. All procedures were implemented in MATLAB 2013a software and executed on an Intel(R) Pentium(R) Dual CPU computer with 2.00GHz of CPU speed and 4.00 GB of RAM.

For the performance measurement of the proposed algorithm, the results of using the selection operator method are compared by solving well-known problems. The proposed algorithm has several parts and the initial solution may be the optimal solution. The results reveal that by the using proposed method, the initial solution will be improved to the optimal or a near-optimal solution in all the test problems. The perfect balance of an assembly line is attained by the combination of work elements in a way that the total busy time of workstations will be the same as cycle time. Since a perfect balance can rarely happen, some other metrics are used in UALBP type-1 to compare different combinations. These metrics can evaluate the performance and efficiency of the balance.

Several workstations (NWS): Having less NWS means more proper task dispatch, which leads to a more efficient balance. Less NWS can save the budgets and working area.

Line Efficiency (*LE*): *LE* is yield by summing up all workstation's time to the *CT* divided by the total workstation number. It reflects the percentage of the line's usage. Higher values of *LE* is more desirable out of an ideal value of one hundred. To maximize *LE* the workstation number must be minimized. *LE* is calculated by equation (8).

$$LE = \left(\frac{\sum_{i=1}^{m} T(S_i)}{M * CT}\right) * 100 \tag{8}$$

where $T(S_i)$ is the total time of tasks that there are in the workstation *i* and calculated by equation (9).

$$T(S_i) = \sum_{j \in S_i} T_j \tag{9}$$

Smoothness Index (*SI*): An important performance variable in a production line. *SI* indicates the total time when a workstation is idle. It usually happens when an improper assignment is done. The ideal value for *SI* is zero, which shows a perfect balance. The minimum value of *SI* can be reached when the differences of the workstations' workloads are decreased as much as possible. *SI* can be computed by equation (10).

$$SI = \sqrt{\frac{\sum_{i=1}^{m} \left(T(S_{max}) - T(S_i)\right)^2}{m}} \tag{10}$$

where $T(S_{max})$ is the maximum value of $T(S_i)$.

This non-linear function given by equation (10) speeds up the transmission of activities from low to the high-pressure workstation. Therefore, the chance of getting an empty workstation in the next solutions will be increased.

Variation (*V*): This is another performance variable of a production line by considering the utilization of each workstation. Similar to the *SI*, the minimum amount of *V* can be accomplished by decreasing the difference of workloads among workstations. *V* is calculated by equation (11).

$$V = \sqrt{\frac{\sum_{i=1}^{m} (U_i - aver)^2}{m}} \tag{11}$$

where $aver$ is an average utilization for all workstations, which is calculated by equation (12). Moreover, the utilization of the workstation, S_i, is calculated by equation (13).

$$aver = \sum_{i=1}^{m} U_i/m \tag{12}$$

$$U_i = T(S_i)/T(S_{max}) \tag{13}$$

3.2. Results Obtained by the Proposed Method

To find the performance level of the proposed method, different types of test problems are considered. For each problem, the number of the tasks, the cycle time, the optimum number of workstations found by Hwang et al. (2008), the number of workstations found by the RPW method, Revised-RPW method, and the proposed GES are reported.

Table 1.Number of workstations

Problem	Number of tasks	Cycle time	The optimum number of workstations	RPW method	Revised-RPW method	GES with Revised-RPW method
Mitchell	21	14	8	9	8	8
		21	5	6	6	5
Heskia	28	138	8	10	8	8
		205	5	6	6	5
Sawyer	30	27	13	15	13	13
		33	10	13	11	10
Tonge	70	176	21	23	21	21
		364	10	11	10	10
Arcus 1	83	6842	12	13	12	12
		8412	10	10	10	10
Arcus 2	111	5755	27	29	27	27
		10743	15	17	15	15

The results in Table 1 show that in all of the considered test problems, the optimum number of workstations obtained by the proposed GES method is the optimal answer.

3.3. Comparing with a GA-Based Method

In Table 2, the test problems with the same cycle time mentioned in Table 1 are considered again. In this table, for the GES methods, two indexes such as *LE*, and *V* are calculated and compared to the obtained solutions proposed GA by Hwang et al. (2008).

Table 2. Comparisons with the GA algorithm

Problem	Number of tasks	Cycle time	GES with Revised-RPW method			Multi-objective GA Fitness function E		
			No. of stations	LE	VI	No. of stations	LE	VI
Mitchell	21	14	8	93.75	0.023	8	93.75	0.055
		21	5	100.0	0.000	5	100.0	0.000
Heskia	28	138	8	92.75	0.006	8	92.70	0.112
		205	5	99.90	0.001	5	99.90	0.001
Sawyer	30	27	13	92.31	0.023	13	92.31	0.071
		33	10	98.18	0.014	10	98.18	0.024
Tonge	70	176	21	94.97	0.042	21	95.00	0.043
		364	10	96.43	0.006	10	96.40	0.023
Arcus 1	83	6842	12	92.26	0.017	12	92.20	0.097
		8412	10	90.22	0.020	10	90.00	0.123
Arcus 2	111	5755	27	96.79	0.019	27	96.79	0.036
		10743	15	93.38	0.017	15	93.38	0.063

According to the above table, the results found by the proposed GES method is better or the same as the reported solutions by the genetic algorithm in the literature.

4. Conclusion

In this research, the assembly line balancing problem considering reducing the number of workstations with a given cycle time is studied. Primarily, a mathematical model and then two new methods including an exact algorithm and a hybrid grouping meta-heuristic algorithm were proposed. The former one was to find an initial solution and the latter one was to improve the initial solution to achieve the best solution, by using a method for selection operator to solve U-shaped assembly line balancing problems. The proposed algorithm is based on the grouping evolution strategies method, while the most useful meta-heuristic algorithm that already exists, is based on the genetic algorithm. Moreover, to increase the performance of the proposed procedure, a modified version of the COMSOAL method was used. The performance of the proposed method was compared to one of the proposed algorithms from the literature. The results illustrated that the proposed algorithm is one of the most efficient algorithms to solve such problems.

Various topics may be considered for possible future studies. For instance, considering pre-assignment of some of the given tasks to especial workstations, and similarly, limitation to assign two or more tasks in the same workstations. Additionally, minimization of cycle time may be considered as the second or simultaneous objective function, which is the other important objective of assembly balancing problems to decrease the working hours. Using a goal programming approach to optimize such problems, or using other meth-heuristic methods like Particle Swarm Optimization, Grey wolf optimization, neural network, etc. may be of interest to researchers in the future. Finally, considering non-deterministic processing time with any of the mentioned objective function may be a good topic to be considered.

References

1. Arcus, A. L. (1965). A computer method of sequencing operations for assembly lines. International Journal of Production Research, 4(4), 259-277.
2. Becker, C., & Scholl, A. (2006). A survey on problems and methods in generalized assembly line balancing. European journal of operational research, 168(3), 694-715.
3. Beyer, H. G., & Schwefel, H. P. (2002). Evolution strategies–a comprehensive introduction. Natural computing, 1(1), 3-52.
4. Boros, P., Fehér, O., Lakner, Z., Niroomand, S., & Vizvári, B. (2016). Modeling supermarket re-layout from the owner’s perspective. *Annals of Operations Research*, *238*(1-2), 27-40.
5. Boysen, N., Fliedner, M., & Scholl, A. (2007). A classification of assembly line balancing problems. European journal of operational research, 183(2), 674-693.
6. Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. IEEE computational intelligence magazine, 1(4), 28-39.
7. Erel, E., Sabuncuoglu, I., & Aksu, B. A. (2001). Balancing of U-type assembly systems using simulated annealing. International Journal of Production Research, 39(13), 3003-3015.
8. Falkenauer, E. (1994). A new representation and operators for genetic algorithms applied to grouping problems. Evolutionary computation, 2(2), 123-144.
9. Fathi, M., Alvarez, M. J., & Rodríguez, V. (2011). A new heuristic approach to solving u-shape assembly line balancing problems type-1. International Journal of Industrial and Manufacturing Engineering, 5(11), 2115-2123.
10. Ghadiri Nejad, M., & Banar, M. (2018). Emergency response time minimization by incorporating ground and aerial transportation. Annals of Optimization Theory and Practice, 1(1), 43-57.
11. Ghadiri Nejad, M., Gueden, H., Vizvari, B., & Vatankhah Barenji, R. (2018). A mathematical model and simulated annealing algorithm for solving the cyclic scheduling problem of a flexible robotic cell. Advances in Mechanical Engineering, 10(1), 1687814017753912.
12. Ghadirinejad, M., & Mosallaeipour, S. (2013). A new approach to optimize a flexible manufacturing cell. In 1st international conference on new directions in business, management, finance and economics (Vol. 38).
13. Ghadirinejad, M., Kashan, A. H., & Rismanchian, F. (2013). A new competitive method for solving assembly line balancing problem. In 1st Int. Conf. New Directions in Business, Management, Finance and Economics.
14. Glonegger, M. and G. Reinhart, Planning of synchronized assembly lines taking into consideration human performance fluctuations. Production Engineering, 2015. 9(2): p. 277-287.
15. Gökçen, H., & Agˇpak, K. (2006). A goal programming approach to simple U-line balancing problem. European journal of operational research, 171(2), 577-585.
16. Gökçen, H., Ağpak, K., Gencer, C., & Kizilkaya, E. (2005). A shortest route formulation of simple U-type assembly line balancing problem. Applied Mathematical Modelling, 29(4), 373-380.

17. Helgeson, W. B., & Birnie, D. P. (1961). Assembly line balancing using the ranked positional weight technique. Journal of industrial engineering, 12(6), 394-398.
18. Hwang, R. K., Katayama, H., & Gen, M. (2008). U-shaped assembly line balancing problem with genetic algorithm. International Journal of Production Research, 46(16), 4637-4649.
19. Jayaswal, S., & Agarwal, P. (2014). Balancing U-shaped assembly lines with resource dependent task times: A Simulated Annealing approach. Journal of Manufacturing Systems, 33(4), 522-534.
20. Kashan, A. H., Jenabi, M., & Kashan, M. H. (2009, December). A new solution approach for grouping problems based on evolution strategies. In 2009 International Conference of Soft Computing and Pattern Recognition (pp. 88-93). IEEE.
21. Kashan, A. H., Kashan, M. H., & Karimiyan, S. (2013). A particle swarm optimizer for grouping problems. Information Sciences, 252, 81-95.
22. Kumar, N., & Mahto, D. (2013). Assembly line balancing: a review of developments and trends in approach to industrial application. Global Journal of Research In Engineering.
23. Li, X., & Zhang, K. (2018). Single batch processing machine scheduling with two-dimensional bin packing constraints. International Journal of Production Economics, 196, 113-121.
24. López-Ibáñez, M., Stützle, T., & Dorigo, M. (2015). Ant Colony Optimization: A Component-Wise Overview.
25. Miltenburg, G. J., & Wijngaard, J. (1994). The U-line line balancing problem. Management science, 40(10), 1378-1388.
26. Miltenburg, J., & Sparling, D. (1995). Optimal solution algorithms for the U-line balancing problem. Relatrio tecnico, McMaster University, Hamilton, Canada. Citado na.
27. Mirzaei, N., Niroomand, S., & Zare, R. (2016). Application of statistical process control in service industry: A case study of the restaurant sector. *Journal of Modelling in Management*.
28. Monden, Y. (2011). Toyota production system: an integrated approach to just-in-time. CRc Press.
29. Mosa, M. A., Hamouda, A., & Marei, M. (2017). Graph coloring and ACO based summarization for social networks. Expert Systems with Applications, 74, 115-126.
30. Mosallaeipour, S., Nazerian, R., & Ghadirinejad, M. (2018). A Two-Phase Optimization Approach for Reducing the Size of the Cutting Problem in the Box-Production Industry: A Case Study. In Industrial Engineering in the Industry 4.0 Era (pp. 63-81). Springer, Cham.
31. Mosallaeipour, S., Nejad, M. G., Shavarani, S. M., & Nazerian, R. (2018). Mobile robot scheduling for cycle time optimization in flow-shop cells, a case study. Production Engineering, 12(1), 83-94.
32. Nadimi-Shahraki, M. H., Taghian, S., & Mirjalili, S. (2021). An improved grey wolf optimizer for solving engineering problems. Expert Systems with Applications, 166, 113917.

33. Nejad, M. G., & Kashan, A. H. (2019). An Effective Grouping Evolution Strategy Algorithm Enhanced with Heuristic Methods for Assembly Line Balancing Problem. Journal of Advanced Manufacturing Systems, 18(03), 487-509.
34. Nejad, M. G., Güden, H., & Vizvári, B. (2019). Time minimization in flexible robotic cells considering intermediate input buffers: a comparative study of three well-known problems. International Journal of Computer Integrated Manufacturing, 32(8), 809-819.
35. Nejad, M. G., Kashan, A. H., & Shavarani, S. M. (2018). A novel competitive hybrid approach based on grouping evolution strategy algorithm for solving U-shaped assembly line balancing problems. Production Engineering, 12(5), 555-566.
36. Nejad, M. G., Kovács, G., Vizvári, B., & Barenji, R. V. (2018). An optimization model for cyclic scheduling problem in flexible robotic cells. The International Journal of Advanced Manufacturing Technology, 95(9), 3863-3873.
37. Nejad, M. G., Shavarani, S. M., Güden, H., & Barenji, R. V. (2019). Process sequencing for a pick-and-place robot in a real-life flexible robotic cell. The International Journal of Advanced Manufacturing Technology, 103(9), 3613-3627.
38. Nejad, M. G., Shavarani, S. M., Vizvári, B., & Barenji, R. V. (2018). Trade-off between process scheduling and production cost in cyclic flexible robotic cells. The International Journal of Advanced Manufacturing Technology, 96(1), 1081-1091.
39. Niroomand, S. (2018). A multi-objective based direct solution approach for linear programming with intuitionistic fuzzy parameters. *Journal of Intelligent & Fuzzy Systems*, *35*(2), 1923-1934.
40. Niroomand, S., Bazyar, A., Alborzi, M., & Mahmoodirad, A. (2018). A hybrid approach for multi-criteria emergency center location problem considering existing emergency centers with interval type data: a case study. *Journal of Ambient Intelligence and Humanized Computing*, *9*(6), 1999-2008.
41. Niroomand, S., Hadi-Vencheh, A., Mirzaei, N., & Molla-Alizadeh-Zavardehi, S. (2016). Hybrid greedy algorithms for fuzzy tardiness/earliness minimisation in a special single machine scheduling problem: case study and generalisation. *International Journal of Computer Integrated Manufacturing*, *29*(8), 870-888.
42. Niroomand, S., Takács, S., & Vizvári, B. (2011). To lay out or not to lay out?. *Annals of Operations Research*, *191*(1), 183-192.
43. Niroomand, S., & Vizvári, B. (2013). A mixed integer linear programming formulation of closed loop layout with exact distances. *Journal of Industrial and Production Engineering*, *30*(3), 190-201.
44. Pereira, J. (2016). Procedures for the bin packing problem with precedence constraints. European Journal of Operational Research, 250(3), 794-806.
45. Pereira, J., Ritt, M., & Vásquez, Ó. C. (2018). A memetic algorithm for the cost-oriented robotic assembly line balancing problem. Computers & Operations Research, 99, 249-261.
46. Rechenberg, I. (1978). Evolutionsstrategien. In Simulationsmethoden in der Medizin und Biologie (pp. 83-114). Springer, Berlin, Heidelberg.

47. Salehi, M., Maleki, H. R., & Niroomand, S. (2020). Solving a new cost-oriented assembly line balancing problem by classical and hybrid meta-heuristic algorithms. *Neural Computing and Applications*, *32*(12), 8217-8243.
48. Sanei, M., Mahmoodirad, A., & Niroomand, S. (2016). Two-stage supply chain network design problem with interval data. *International Journal of e-Navigation and Maritime Economy*, *5*, 74-84.
49. Scholl, A., & Becker, C. (2006). State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. European Journal of Operational Research, 168(3), 666-693.
50. Scholl, A., & Klein, R. (1999). ULINO: Optimally balancing U-shaped JIT assembly lines. International Journal of Production Research, 37(4), 721-736.
51. Shavarani, S. M., Nejad, M. G., Rismanchian, F., & Izbirak, G. (2018). Application of hierarchical facility location problem for optimization of a drone delivery system: a case study of Amazon prime air in the city of San Francisco. The International Journal of Advanced Manufacturing Technology, 95(9), 3141-3153.
52. Taassori, M., Taassori, M., Niroomand, S., Vizvári, B., Uysal, S., & Hadi-Vencheh, A. (2015). OPAIC: An optimization technique to improve energy consumption and performance in application specific network on chips. *Measurement*, *74*, 208-220.
53. Tavana, M., Santos-Arteaga, F. J., Mahmoodirad, A., Niroomand, S., & Sanei, M. (2018). Multi-stage supply chain network solution methods: hybrid metaheuristics and performance measurement. *International Journal of Systems Science: Operations & Logistics*, *5*(4), 356-373.
54. Toklu, B., & özcan, U. (2008). A fuzzy goal programming model for the simple U-line balancing problem with multiple objectives. Engineering Optimization, 40(3), 191-204.
55. Vatankhah Barenji, R., Ghadiri Nejad, M., & Asghari, I. (2018). Optimally sized design of a wind/photovoltaic/fuel cell off-grid hybrid energy system by modified-gray wolf optimization algorithm. Energy & Environment, 29(6), 1053-1070.
56. Vizvari, B., Guden, H., & G Nejad, M. (2018). Local search based meta-heuristic algorithms for optimizing the cyclic flexible manufacturing cell problem. Annals of Optimization Theory and Practice, 1(3), 15-32.
57. Yuan, L., Qin, L., Lin, X., Chang, L., & Zhang, W. (2017). Effective and efficient dynamic graph coloring. Proceedings of the VLDB Endowment, 11(3), 338-351.

Published by Firouzabad Institute of Higher Education, Firouzabad, Fars, Iran