Production Engineering (2018) 12:555–566
https://doi.org/10.1007/s11740-018-0836-x

ASSEMBLY

A novel competitive hybrid approach based on grouping evolution strategy algorithm for solving U-shaped assembly line balancing problems

Mazyar Ghadiri Nejad[1] · Ali Husseinzadeh Kashan[2] · Seyed Mahdi Shavarani[1]

Received: 22 February 2018 / Accepted: 29 May 2018 / Published online: 7 June 2018

Abstract

Assembly line balancing problems (ALBPs) are among the well-known problems in manufacturing systems that belong to NP-hard class of problems. In the literature, there are various metaheuristic methods proposed to solve different models of such a problem under various assumptions. This research considers the U-shaped ALBP and proposes a hybrid solution method based on grouping evolution strategy algorithm. To develop a competitive approach, two most popular constructive methods of solving ALBP including the ranked positional weight method, and COMSOAL algorithm are modified and improved. We investigate the effectiveness of the proposed improvements and evaluate the performance of the proposed approach via solving a number of existing problems in the literature and compare the results with some current methods in the literature. Computational results indicate that the proposed approach for solving U-shaped ALBP test problems performs efficiently and is able to obtain the global optimal solution of the most of high dimensional problems.

Keywords U-shaped assembly line balancing · Grouping problems · Heuristics · Metaheuristics · Grouping evolution strategy

1 Introduction

An assembly line is defined as a number of arranged workstations where the components of a particular product are attached to each other to make finished goods. Assembly line balancing (ALB) is described as changing the arrangement of activities or tasks in the workstations upon some specific criteria to gain the optimum performance/throughput [22]. The fundamental ALB problem can be described mathematically as follows:

$$\min z = \sum_{i=1}^{M} \max_{j=1,\dots,N}\{x_{ij}\} \tag{1}$$

$$\sum_{i=1}^{M} x_{ij} = 1 \quad \forall j = 1,\dots,N \tag{2}$$

$$\sum_{i=1}^{M} ix_{ik} \leqslant \sum_{i=1}^{M} ix_{ij} \quad \forall j = 1,\dots,N, \quad k \in Precedence(j) \tag{3}$$

$$\sum_{j=1}^{N} t_j x_{ij} \leqslant CT \quad \forall i = 1,\dots,M \tag{4}$$

$$x_{ij} = 0,1 \quad \forall\, i = 1,\dots,M, \quad j = 1,\dots,N \tag{5}$$

Objective (1) minimizes the number of workstations. The first constraint set assigns each task to exactly one station. Constraints set (3) forces the precedence relationship between tasks. Constraints set (4) considers an upper bound on the cycle time of each workstation.

The straight assembly lines are a serial arrangement of workstations in a line. The straight and U-shaped layouts are two most popular layouts in production and assembly

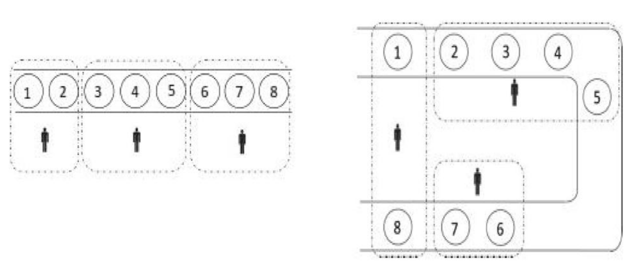
✉ Mazyar Ghadiri Nejad
mazyar.ghadirinejad@gau.edu.tr

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

[2] Department of Industrial and Systems Engineering, Tarbiat Modares University, Tehran, Iran

lines. A number of inefficiencies have been addressed in the literature on the line flexibility, job monotony and large inventories for the straight ALB [3]. After introducing the Just-in-Time production system, U-shaped assembly lines became more popular. Figure 1a, b, show the examples of straight and U-shaped assembly line layouts for eight tasks and three workstations, respectively. In U-shaped arrangements, the entrance and discharging points are set to the end of U (see the left end of the line in Fig. 1b). This type of layout lets the operating personnel work on both fronts in a given cycle. The workstations included in both sides of the line are called crossover workstations (see the station at the left end of Fig. 1b where the personnel is working on both task 1 and 8, simultaneously). If the number of crossover workstations increases, the flexibility of task-workstation combinations increases and consequently, there will be more efficient balance by a less number of workstations and operating personnel [11]. Productivity improvement, reduction in work-in-process inventory, space requirement, and lead-time are the other benefits of U-shaped assembly lines [28].

In the last decades, various studies have been carried out to handle large-size U-shaped assembly line balancing problems (UALBP), mainly when the objective is to reduce the number of workstations for a given cycle time [21]. ULINO (U-line optimizer), proposed by Scholl and Klein [29] is a branch-and-bound procedure that performs a depth-first search using bounds and some dominance rules to solve different versions of UALBP. Hwang et al. [18] proposed a genetic algorithm solution method to solve a multi-objective UALBPs. They considered principles of just-in-time production since the UALB systems have more benefits in comparison with the straight ALB systems. They tried to minimize the number of workstations and variation of workloads simultaneously. Jonnalagedda and Dabade [19] suggested a genetic algorithm for solving UALBPs for minimizing cycle time and maximizing the line efficiency index under a given number of workstations. Combined advantages of parallel assembly line system and UALB have been demonstrated by Kucukkoc and Zhang [20], where their aim was maximizing resource utilization. Ogan and Azizoglu [25] proposed a mixed integer linear formulation and a branch and bound algorithm to solve the UALBPs considering equipment cost of assigning the tasks to the workstations. Mukund Nilakantan and Ponnambalam [24] developed an algorithm based on particle swarm optimization approach to solve a robotic UALBP and illustrated that the efficiency of the UALB is better than the straight ALB. Recently, Oksuz et al. [26] developed linear and non-linear mathematical formulations for UALBPs aiming to maximize the line efficiency index and considering labors' performances. Additionally, they proposed two metaheuristic algorithms based on genetic and artificial bee colony algorithm in their research.



(a) Straight assembly line **(b)** U-Shaped assembly line

Fig. 1 Assembly line layouts

In division and grouping allocation problems like ALBP [9], clustering problem [2], maximally diverse grouping problem [5], and graph coloring problem [31], metaheuristic algorithms which work based on group structure, i.e., grouping genetic algorithm, are more efficient (Falkenauer and Delchambre [6]). Grouping problems are concerned with partitioning a set of objects into a collection of disjoint subsets such that the union of the subsets constructs the whole set of the objects [17]. Grouping problems usually contain a set of constraints that must be satisfied in the task-assignments. It means, not all assignments are acceptable. Grouping problems include an objective function upon a different combination of the groups. Using evolutionary algorithms, a group/subgroup must be held as a block in the course of search. Based on this fact, researchers have used evolutionary algorithms to improve the quality of solving grouping problems [16, 17].

Grouping evolution strategy (GES) is a kind of evolutionary algorithm that is proposed for grouping problems. Before GES, the grouping genetic algorithm (GGA) was the most predominant algorithm for grouping problems which uses a particular type of representation (grouping representation) and operators for grouping problems. For details, the interested reader may refer to Falkenauer and Delchambre [6]. Introduced by Husseinzadeh Kashan et al. [15], GES is compatible with evolution strategy (ES) of Rechenberg [27] with this distinction that ES uses Gaussian mutation during optimization process whereas GES benefits from a novel comparable mutation operator working based on the rationale of a two-phase dropping and adding strategy suitable for grouping problems (e.g., ALBP) under grouping representation. It has been proven that GES has merit for solving grouping problems since it owns some unique characteristics that GGA cannot afford. Some successful applications of GES have been reported on bin packing problem, batch processing problem, fuzzy data clustering, parallel machine scheduling problem, helicopter routing problem etc [1, 9, 14, 16, 17]. Following the successful applications of GES on grouping problems and using the Hwang et al. study [18], we propose a hybrid method to minimize the number of workstations for a given cycle time in the U-shaped ALBP.

To start GES with a good initial seed solution, we inspire from the ranked positional weight (RPW) method [7], for generating a feasible assignment of tasks to workstations. In this way, to increase the chance of starting GES with a good seed and to obtain probably better results as initial solutions from the objective function point of view, the modified version of RPW method named Revised-RPW is developed and utilized. To prove the superiority of Revised-RPW over the traditional RPW method, we use from the assumption of Fathi et al. [7] in addition to their precedence diagram. To construct a complete and feasible offspring as the output of the mutation phase of GES, a modified version of the COMSOAL algorithm presented by Arcus [4] is developed. To test the performance of the proposed method, some well-known standard test problems are employed.

This paper is organized as follows: in the next section, the overall architecture of the hybrid GES algorithm is proposed. Section 3 includes experimental results carried out on small and large size UALB test problem instances. Finally, Sect. 4 contains discussions and concludes the paper.

2 The proposed hybrid GES algorithm for UALBP

In this section, a hybrid algorithm is proposed to minimize the number of workstations for a given cycle time for UALBPs. Our approach utilizes the structural information of the problem along with randomness. The randomness provides a mechanism to scape local optima, but at the same time, the final output may vary from one run to another even though the algorithm parameters keep unchanged. In the following, the details of the proposed solution method for solving the mentioned UALBP are presented. The proposed approach composes of the following four-states:

1. Using a constructive algorithm based on Revised-RPW method to generate an initial solution
2. Using a heuristic method based on Revised-COMSOAL method for assigning the missed activities during mutation stage of GES.
3. Using the GES metaheuristic algorithm to generate new solutions
4. Using a selection method to select fitter individuals in the course of GES search process.

2.1 Generating an initial solution: a modified heuristic method

There are several methods for determining an initial solution for a UALBP including heuristic methods with their strengths and weaknesses. To create an initial solution, at first RPW method proposed by Helgeson and Birnie [13] and its modified version for UALBP proposed by Fathi et al. [7] is considered. Based on the modifications of Fathi et al. method, a number of changes are made on the classical RPW method to improve its output. We call it the Revised-RPW method. In the following, the RPW method and the Revised-RPW method are explained in details, and their quality of balancing are compared with each other using performance indicators.

Based on the RPW method for UALBP, the positional weight of each task in a U-shaped layout should be determined in both forward and backward directions. In the forward direction, the positional weight of each task is the total time from that task to the last task in the precedence graph in the longest path. The positional weight for the backward direction is calculated similarly but in the opposite direction. The positional weight of each task is its larger weight earned by the forward and backward calculations. Then the tasks are sorted in descending order based on their positional weights. There are two criteria for assigning tasks to workstations: First, succession and precedence priorities must be held. It must be mentioned that in UALBPs, the tasks with no predecessor from the beginning of the precedence diagram, and the tasks with no successors from the end of the precedence diagram, are the candidates to be assigned to the first workstation (please see Fig. 1b). Second, the workstation must have idle time to handle the assigning task. In case of multiple available tasks, the one with the highest weight is selected and assigned. The assignment will be completed when there is no task left on the list. To explain how the RPW method solves a UALBP, let's consider the precedence diagram in Fig. 2.

Figure 2, illustrates 12 nodes resembling the tasks with their processing times demonstrated on the top of the node, and the cycle time which is considered to be 11 time units as an example. Table 1 shows the forward, backward and the ranked positional weight of the given precedence diagram.

Utilizing RPW method, in the first iteration, tasks 1 and 12 are the candidates for assignment to the first workstation since their positional weight is maximum. Let us choose task 1 randomly and assign it to workstation 1. In next iteration, between available tasks in the candidate list, task 12 is selected. Since its task time is more than the idle time of the last workstation (workstation 1), it is assigned to a new

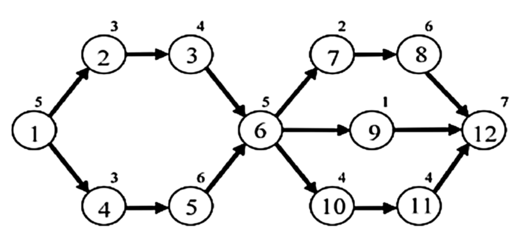


Fig. 2 A precedence diagram

Table 1 Computing the RPW for the tasks given in Fig. 2

Task number	1	2	3	4	5	6	7	8	9	10	11	12
Forward positional weight	5	8	12	8	14	19	21	27	20	23	27	34
Backward positional weight	34	27	24	29	26	20	15	13	8	15	11	7
Ranked positional weight	34	27	24	29	26	20	21	27	20	23	27	34

Table 2 The solution of UALBP using RPW method

Step	Candidate list	Assigned task	Station no.	Station's idle time
1	1, 12	1	1	6
2	2, 4, 12	12	2	4
3	2, 4, 8, 9, 11	4	2	1
4	2, 5, 8, 9, 11	2	3	8
5	3, 5, 8, 9, 11	8	3	2
6	3, 5, 7, 9, 11	11	4	7
7	3, 5, 7, 9, 10	5	4	1
8	3, 7, 9, 10	3	5	7
9	6, 7, 9, 10	10	5	3
10	6, 7, 9	7	5	1
11	9, 6	6	6	6
12	9	9	6	5

Table 3 The solution of UALBP using Revised-RPW method

Step	Candidate list	Assigned task	Station no.	Station's idle time
1	1, 12	1	1	6
2	2, 4, 12	12	2	4
3	2, 4, 8, 9, 11	4	1	3
4	2, 5, 8, 9, 11	2	1	0
5	3, 5, 8, 9, 11	8	3	5
6	3, 5, 7, 9, 11	11	2	0
7	3, 5, 7, 9, 10	5	3	0
8	3, 7, 9, 10	3	4	7
9	6, 7, 9, 10	10	4	3
10	6, 7, 9	7	4	1
11	9, 6	6	5	6
12	9	9	4	0

workstation. Following the steps of RPW method, all tasks are assigned to 6 workstations (see Table 2).

To possibly reduce the number of workstations, the RPW method is modified in a way that each task is assigned to the last workstation that contains any of its predecessor activities, or to one of the next workstations. Similarly, each activity is assigned to the last workstation that contains any of its successor activities, or to one of the next workstations. Table 3 shows the results of using the modified method.

Table 4 Comparing RPW and Revised-RPW Method

Solution method	Work-station number	Line efficiency (%)	Smoothness	Variation
RPW	6	83.33	6.4807	0.2055
Revised-RPW	5	90.91	4.1231	0.1408

The first two steps of Revised-RPW are quite similar to the RPW solution. In the third step where task 4 is the candidate task, considering its processing time, it can be assigned to workstation 1 based on the new rule, because its precedence which is task 1, has been assigned to workstation 1. Therefore, task 4 is assigned to the first workstation. By continuing the same procedure, the results in Table 3 are earned. To compare Revised-RPW versus RPW, four performance indicators are considered as follows [12]:

1. *Number of workstations*
2. *Line efficiency* (LE) index which is equal to $\left(\frac{\sum_{i=1}^{m} T(S_i)}{m \times CT}\right) \times 100$.
3. *Smoothness index* which is the standard deviation of work distribution between the workstations, and is equal to $\sqrt{\frac{\sum_{i=1}^{m} \left(T(S_{max}) - T(S_i)\right)^2}{m}}$.
4. *Variation* which determines the standard deviation of workstation utilization, and is calculated as $V = \sqrt{\frac{\sum_{i=1}^{m} (U_i - aver)^2}{m}}$.

where S_i addresses workstation i, $T(S_i)$ is the total processing times of the tasks assigned to workstation S_i, m is the number of workstations, CT is the cycle time, TS_{max} is the maximum value among the workstations total time, U_i is the utilization ratio of workstation i, which is equal to $U_i = T(S_i)/T\left(S_{max}\right)$, and $aver = \sum_{i=1}^{m} U_i/m$ [18]. Based on the ideal ALB objectives, the number of workstations, smoothness and variation indexes should be minimized, and the line efficiency index should be maximized.

The results in Table 4 show that the number of workstations, the smoothness index, and the variation index found by the Revised-RPW method is less, and the line efficiency index found by this method is higher than the classical RPW method. Therefore, the Revised-RPW method is superior to traditional RPW method.

2.2 The grouping evolution strategy (GES) algorithm for UALBP

Since ALBP belongs to non-deterministic polynomial-time (NP-hard) class of problems [3], exact algorithms may only give the optimal solutions for small-sized problem instances. To solve large-sized problems, metaheuristic algorithms can be utilized. In this regard, based on evolution strategies algorithm, a grouping evolution strategy (GES) algorithm is developed for UALBP in this study.

Rechenberg [27] proposed ES algorithm which is a mathematical formulation of Darwinian biological evolution and utilized it as a general optimization technique. In each generation of ES, a set of solutions (offspring) are produced from the existing solutions (parents) via recombination and mutation operators. For recombination, a number of parents are taken randomly, and their centroid point is calculated. A symmetric point perturbation is added to the recombination output to generate minor deviations. For mutation, the perturbation is chosen from an isotropic normal distribution. Selection in ES can be either among the last parents and the new offspring or only among new offspring. The grouping evolution strategy is a new variant of evolution strategy developed for grouping problems which are discrete.

One of the issues to design a metaheuristic algorithm is the solution representation [8, 10, 23, 30]. To represent a solution of UALBP by GES, a structure whose length is equal to the number of workstations is considered, wherein each element of structure which is associated to a workstation includes a set of tasks. Figure 3, demonstrates the structure associated with the solution of the Revised-RPW method for the given diagram in Fig. 2. In this balancing scheme, there are five workstations where tasks 1, 3 and 4 are assigned to workstation 1, tasks 2 and 6 are assigned to workstation 2 and so forth. Since the workstations make role as groups, adopting such a structure is called grouping encoding, and GES works with the sets of $\{T_1, T_3, T_4\}$, $\{T_2, T_6\}$, $\{T_5, T_7\}$, $\{T_8, T_9, T_{10}, T_{12}\}$, $\{T_{11}\}$ as a chromosomal structure with five genes, including one gene for each workstation.

$$\{\overset{S_1}{\{T_1, T_3, T_4\}}, \overset{S_2}{\{T_2, T_6\}}, \overset{S_3}{\{T_5, T_7\}}, \overset{S_4}{\{T_8, T_9, T_{10}, T_{12}\}}, \overset{S_5}{\{T_{11}\}}\}$$

Fig. 3 A solution representation

2.3 Mutation operator and a constructive heuristic based on COMSOAL algorithm to generate new solutions in GES

To prevent having static solutions, the other solutions except the initial solution, are generated based on GES mutation operator. The logic of GES mutation operator is removing a number of the assigned tasks from workstations of a given parent solution, to obtain an incomplete solution. Thereafter, a constructive heuristic method is utilized to assign the missed tasks to the current or newly opened workstations. For more details about the mutation operator in GES, the reader is suggested to refer to Husseinzadeh Kashan et al. [15].

The Revised-RPW cannot be used for reassigning the removed tasks to workstations. Otherwise, the same solution would be obtained. Instead due to its simplicity and flexibility, COMSOAL method [4] is considered for reassignment. One of the strengths of this approach is its ability to produce different solutions resulted by the random selection process during the allocation of the tasks to workstations in each step. The procedure of the COMSOAL method to solve the UALBP for the given precedence graph in Fig. 2, includes the following steps:

Step 1 Find the minimum between the number of predecessors and successors of each task (see Table 5). In this step, at first the number of predecessors of each task is counted. For example, the predecessor number of task 1 is zero, since this task does not have any predecessor, or the predecessor number of task 2 is one, because this task can be performed directly after task one (please see the second row of Table 5). Then, the number of successors of each task must be counted (please see the third row of Table 5). At the end, the minimum number of the predecessors and successors must be found for each task (please see the last row of Table 5).

Step 2 Randomly select one of the tasks with zero minimum number and assign it to the last existing or newly opened workstation based on the task processing time and workstation's idle time (for instance select task 1 between task 1 and 12 and assign it to the first workstation).

Step 3 Update Table 5 after removing the assigned task and go to Step 2 if there is any unassigned task (see Table 6).

To utilize the COMSOAL method for reassignment of the removed tasks during the mutation operator, and to improve the solutions, two modification strategies are considered. The COMSOAL method always starts from the first task (node) and continues forward by selecting only one task among unassigned tasks, in each step. Then it assigns the chosen task to

Table 5 Number of predecessors and successors

Task number	1	2	3	4	5	6	7	8	9	10	11	12
Number of predecessors	0	1	1	1	1	2	1	1	1	1	1	3
Number of successors	2	1	1	1	1	3	1	1	1	1	1	0
The minimum number	0	1	1	1	1	2	1	1	1	1	1	0

Table 6 Updated table for the number of the predecessors and successors

Task number	2	3	4	5	6	7	8	9	10	11	12
Number of predecessors	0	1	0	1	2	1	1	1	1	1	3
Number of successors	1	1	1	1	3	1	1	1	1	1	0
The minimum number	0	1	0	1	2	1	1	1	1	1	0

the last or newly opened workstation considering the idle time of the last workstation. In other words, the COMSOAL method always deals with the tasks from both ends of the precedence graph. Whereas in the current study, after applying the mutation operator, some tasks in the middle of the precedence diagram may be required to be reassigned. Therefore, at first, those tasks with no predecessor or successors are distinguished and then, based on their assigned predecessors/successors the reassignments are applied. Additionally, to enhance the quality of the solutions, the Critical Path (CP) of the precedence graph is found, and the tasks belonging to the CP are first reassigned to the workstations. Such a modified version of COMSOAL method, named as "CP-COMSOAL" method, is presented as follows:

Step 1 Determine the CP of the given precedence diagram.

Step 2 Among the unassigned tasks find the ones with no predecessor or successor. If there is more than one task, select the task belonging to the CP. If there is more than one task belonging to CP, select randomly. If none of the tasks belong to the CP, choose one task randomly.

Step 3 Assign the selected task to a workstation, considering its predecessors and successors, based on the idle times of existing workstations. The selected task, because of having no predecessor in the partial solution, cannot be assigned to a workstation which lies before the ones that contain its predecessors. Similarly, the selected task because of having no successor cannot be assigned to a workstation after the ones that contain its successors. If there is no feasible workstation to assign the selected task, a new workstation is opened.

Step 4 Repeat Step 2, if there is any task which has not been assigned yet.

2.4 Selecting the best solution in each iteration of GES

In each iteration of GES, the best solution should be chosen to enter into the next iteration. In this regard, all the three mentioned indicators, i.e., line efficiency (LE), smoothness index (SI) and variation index (VI), are recalled. To find the best solution, at first their number of the workstations are compared, and the least one is selected. If there is more than one solution with the least number of the workstations, the one with the highest LE is selected. In case of having at least two solutions with the same number of workstation and LE value, the one with the least VI value is selected. Finally, the least SI value is considered when all the three mentioned indicators are equal.

In what follows, we provide the algorithmic pseudo code of the proposed GES algorithm. For definition of the input parameters of the algorithm the reader may refer to Husseinzadeh Kashan et al. [14].

Algorithm (1+λ)-GES
Initial: $\beta, \lambda, 0 < a \leq 1, \alpha^0 > 0, \alpha^{min} > 0, G \geq 1, P_s$;
Begin
$t \leftarrow 0; G_s \leftarrow 0;\ \alpha \leftarrow \alpha^0$;
Generate an initial feasible solution using Revised-RPW X^t *and evaluate it*;
While stopping criteria are not true
  For $i = 1\ to\ \lambda$
    Given the parent solution X^t , apply CP-COMSOAL algorithm to obtain the offspring solution Y_i^t;
  End for
  Apply the comparison criteria, based on the selection mechanism explained in Section 4.2, between X^t and the λ generated offspring to select the best
  Individual, which is known as X^{t+1};
  If $f(X^{t+1}) < f(X^t)$
    $G_s \leftarrow G_s + 1$;
  End if
  If $(t\ mod\ G) = 0$
    $$\alpha \leftarrow \begin{cases} \alpha/a & if\ G_s/G \geq P_s \\ \max(\alpha_{min}, a \times \alpha) & if G_s/G < P_s \end{cases}$$
    $G_s \leftarrow 0$;
  End if
  $t \leftarrow t + 1$;
  $\alpha^t \leftarrow \alpha$;
End While
End

3 Results and discussions

In this section, we consider six different well-known problems such as Mitchell, Heskia, Sawyer, Tonge, Arcus1, and Arcus2, with various cycle times. The results obtained by the mentioned methods in this study are compared together, and compared with the best solutions achieved by the proposed method of Hwang et al. [18]. These problem instances include the precedence diagrams, cycle times and the optimal solutions that are available at http://www.assembly-line-balancing.de. All methods were implemented in MATLAB 2017a software and executed on an Intel(R) Core(TM) i5-3320 CPU @ 2.60 GHz with 4.0 GB of RAM.

3.1 Comparisons between using the RPW and Revised-RPW in the proposed GES algorithm

In this part, we prepare some comparisons and discussions about the impact of using the classical RPW method or Revised-RPW method, as the initial solution generators in our proposed GES method.

Table 7, presents the objective function values obtained by the RPW, Revised-RPW, and proposed GES method by using each of the RPW or Revised-RPW methods as the initial solution generator, separately for different problems. The first two columns of this table show the name and the tasks number of the problems, and the third column illustrates the corresponding cycle times. For each problem, the optimum number of workstations obtained by Hwang et al. [18] has been reported in the fourth column (Optimum OBF). The obtained objective function (OBF) values using the RPW or Revised-RPW methods are mentioned in the fifth and sixth columns, respectively. The objective function values and CPU times of the proposed GES algorithm using RPW, and Revised-RPW methods as the initial solutions, are placed in the last columns of the table. To solve these problems by using the proposed GES algorithm, we let the algorithm to continue its process to find the optimal solution. Therefore, there was no limit to stop the computations processes except finding the optimal objective function. The reported results in the table are the average of 10 times execution of this algorithm for each test problem.

Table 7 Results of the problems by RPW, Revised-RPW, and the proposed GES methods

Problem name	Number of tasks	Cycle time	Optimum OBF	RPW OBF	Revised-RPW OBF	Initial solution by RPW		Initial solution by Revised-RPW	
						OBF	CPU time	OBF	CPU time
Mitchell	21	14	8	9	8	8	0.515	8	0.608
		15	8	9	8	8	0.078	8	0.031
		21	5	6	6	5	1.327	5	1.310
Heskia	28	138	8	10	8	8	0.078	8	0.156
		205	5	6	6	5	16.895	5	8.284
		324	4	4	4	4	0.0156	4	0.031
Sawyer	30	27	13	15	13	13	0.436	13	0.249
		33	10	13	11	10	6.835	10	8.784
		54	6	7	6	6	14.268	6	0.445
Tonge	70	176	21	23	21	21	37.527	21	0.655
		364	10	11	10	10	52.946	10	0.668
		468	8	8	8	8	0.265	8	0.249
Arcus1	83	5853	13	15	14	13	41.583	13	23.868
		6842	12	13	12	12	10.795	12	0.358
		8412	10	10	10	10	0.171	10	0.156
		10,816	8	8	8	8	0.130	8	0.141
Arcus2	111	5755	27	29	27	27	69.128	27	0.294
		10,027	16	17	16	16	51.627	16	0.796
		10,743	15	17	15	15	1.9968	15	0.667
		17,067	9	10	9	9	43.284	9	0.443

Table 8 Results of the proposed GES by using RPW and Revised-RPW as the initial solution

Problem name	Solution method	Indicator	Best	Worst	Average ± 1-sigma
Tonge	RPW + GES	Objective function	21	23	21.773 ± 0.422
		Line efficiency (%)	94.967	86.709	91.285 ± 0.016
		Smoothness index	47.592	203.934	108.422 ± 19.139
		Variation index	0.037	0.2065	0.075 ± 0.015
		CPU time	14.031	18.953	15.227 ± 0.597
	Revised-RPW + GES	Objective function	21	21	21 ± 0.000
		Line efficiency (%)	94.967	94.967	94.967 ± 0.000
		Smoothness index	46.765	84.486	60.631 ± 5.471
		Variation index	0.037	0.091	0.055 ± 0.009
		CPU time	13.751	24.062	16.551 ± 1.154
Arcus1	RPW + GES	Objective function	12	13	12.991 ± 0.094
		Line efficiency (%)	92.208	85.115	90.179 ± 0.006
		Smoothness index	600.449	4152.118	2584.520 ± 473.852
		Variation index	0.016	0.132	0.065 ± 0.016
		CPU time	9.031	13.437	9.693 ± 0.506
	Revised-RPW + GES	Objective function	12	12	12 ± 0.000
		Line efficiency (%)	92.208	92.208	92.208 ± 0.000
		Smoothness index	525.106	2626.636	1857.027 ± 243.948
		Variation index	0.011	0.083	0.036 ± 0.008
		CPU time	9.453	13.937	10.340 ± 0.634

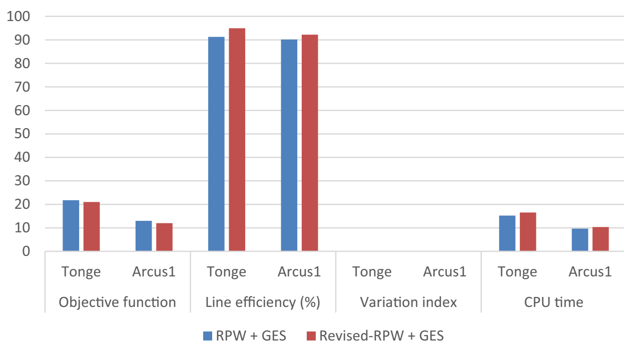


Fig. 4 Comparison between the values of the indicators

According to the results of column entitled “RPW OBF” in Table 7, the classical RPW method could find the optimal solutions of only four problems, in comparison with the results of the Revised-RPW method (column entitled “Revised-RPW OBF”) which could find the optimal solutions of 16 problems out of the 20 mentioned problem instances. The results of these two columns show that the performance of the Revised-RPW method is much better than the performance of the classical RPW method.

The results of using RPW and Revised-RPW as the initial solution for the proposed GES algorithm, show that the proposed GES algorithm finds the optimal solutions of all the problem instances. It means from finding the optimum solutions; there is no difference between using each of the initial solution generators. Considering the computational times for using both methods in the proposed GES, it is clear that the CPU time of the GES by using the Revised-RPW method is less than the CPU times of the using the RPW method. Additionally, using each of RPW and Revised-RPW methods as a source for generating the initial solution, and giving more time to the proposed GES method to search for optimal solution, the chance of finding the optimal solution is increased but, to obtain a competitive methodology for solving the UALBPs, we are obligated to shorten the computation time, or limit the number of iterations to get the optimal (or near optimal) solutions. Hence, in the next section, we compare these methods in a competitive condition and discuss the distribution of the results.

3.2 Distribution comparisons of the methods in 1000 runs

To show the performance quality of the proposed Revised-RPW method and having a fair comparison with the RPW method, in this section, we are going to select two standard problems among large-sized standard problems, and report the statistics on all the mentioned indicators including line efficiency index, smoothness index, variation index, and

Table 9 LE, SI, and VI of the problem instances

Problem	Number of tasks	Cycle time	RPW method			Revised-RPW method			Proposed GES method		
			LE	VI	SI	LE	VI	SI	LE	VI	SI
Mitchell	21	14	83.33	0.066	9.644	93.75	0.097	4.583	93.75	0.023	2.823
		15	83.33	0.158	9.644	87.50	0.161	8.660	87.50	0.023	3.095
		21	83.33	0.171	1.288	83.33	0.311	18.139	100.0	0.000	0.000
Heskia	28	138	76.42	0.187	127.507	92.75	0.100	48.249	92.75	0.006	13.570
		205	83.66	0.196	127.366	83.66	0.366	202.015	99.90	0.001	1.000
		324	80.50	0.248	200.415	80.50	0.364	272.000	97.01	0.022	37.660
Sawyer	30	27	80.00	0.155	36.476	92.31	0.091	11.619	92.31	0.023	8.300
		33	75.52	0.186	36.538	89.26	0.169	21.886	98.18	0.014	3.464
		54	85.71	0.161	30.757	100.0	0.000	0.000	100.0	0.000	0.000
Tonge	70	176	76.70	0.165	267.806	94.97	0.093	85.018	94.97	0.037	46.765
		364	89.63	0.067	147.587	96.43	0.045	84.876	96.43	0.006	22.567
		468	93.80	0.042	72.512	93.80	0.134	196.000	93.80	0.004	37.581
Arcus1	83	5853	86.51	0.192	5296.6	92.40	0.106	2762.1	92.40	0.013	969.803
		6842	86.35	0.142	4796.1	92.21	0.200	5074.1	92.21	0.011	525.106
		8412	90.22	0.155	4624.7	90.22	0.241	6909.9	90.22	0.020	1557.869
		10,816	87.55	0.201	7231.9	87.55	0.295	9782.2	87.55	0.015	1540.256
Arcus2	111	5755	79.36	0.173	8881.2	96.79	0.067	2232.0	96.79	0.019	1545.900
		10,027	88.60	0.080	5718.6	93.76	0.211	8822.3	93.76	0.018	2230.803
		10,743	82.71	0.202	7431.1	93.38	0.187	8247.7	93.38	0.017	2182.117
		17,067	90.01	0.088	7042.3	97.93	0.052	2841.3	97.93	0.004	643.820
Average			84.16	0.152	2604.5	91.62	0.1645	2381.2	94.54	0.0138	568.625

Table 10 Comparisons between the results of the proposed GES method and GA algorithm

Problem	Number of tasks	Cycle time	Proposed GES method			Multi-objective GA Fitness function E			Multi-objective GA Fitness function EV		
			OBF	LE	VI	OBF	LE	VI	OBF	LE	VI
Mitchell	21	14	8	93.75	0.023	8	93.75	0.055	8	93.75	0.023
		15	8	87.50	0.023	8	87.50	0.139	8	87.50	0.023
		21	5	100.0	0.000	5	100.0	0.000	5	100.0	0.000
Heskia	28	138	8	92.75	0.006	8	92.70	0.112	8	92.70	0.007
		205	5	99.90	0.001	5	99.90	0.001	5	99.90	0.001
		324	4	97.01	0.022	4	97.01	0.332	4	97.01	0.062
Sawyer	30	27	13	92.31	0.023	13	92.31	0.071	13	92.31	0.023
		33	10	98.18	0.014	10	98.18	0.024	10	98.18	0.014
		54	6	100.0	0.000	6	100.0	0.000	6	100.0	0.000
Tonge	70	176	21	94.97	0.037	21	95.00	0.043	21	95.00	0.029
		364	10	96.43	0.006	10	96.40	0.023	10	96.40	0.009
		468	8	93.80	0.004	8	93.80	0.063	8	93.80	0.004
Arcus1	83	5853	13	92.39	0.013	13	92.39	0.061	13	92.39	0.014
		6842	12	92.21	0.011	12	92.20	0.097	12	92.20	0.019
		8412	10	90.22	0.020	10	90.00	0.123	10	90.00	0.038
		10,816	8	87.55	0.015	8	87.50	0.200	8	87.50	0.089
Arcus2	111	5755	27	96.79	0.019	27	96.79	0.036	27	96.79	0.019
		10,027	16	93.76	0.018	16	93.70	0.074	16	93.70	0.029
		10,743	15	93.38	0.017	15	93.30	0.063	15	93.30	0.031
		17,067	9	97.93	0.004	9	97.90	0.019	9	97.90	0.004

CPU time for the proposed GES algorithm. Therefore, at first two problem instances, namely the Tonge problem with a cycle time of 176 and a minimum number of workstations equal to 21, and Arcus1 problem with a cycle time of 6842 and optimal objective function of 12 are considered. We use these two problems since they are well recognized problem instances of UALBP. Then, the proposed GES algorithm are relaxed to process 1000 iterations to find the best possible solutions, and 1000 individual runs are performed to compare the averages and the standard deviations of all indicators. Table 8 indicates the results such as the best, the averages, and standard deviations among 1000 runs on each problem. According to the nature of the performance indicators, the maximum value of LE, and the minimum values for SI, VI and CPU time are favorable.

According to the results shown in Table 8, the proposed GES algorithm using the Revised-RPW method for generating an initial solution finds the optimal objective function of both problem instances in all of the 1000 runs with line efficiency values of 94.976 and 92.208%, respectively. However, the GES algorithm initialized with RPW method is not able to always find the optimal solutions for both cases.

Figure 4 illustrates the comparison between the values of the indicators for both methods considering their average results in Table 8.

The bar charts related to each indicator for each problem illustrate the smaller values for objective function, variation index, and higher line efficiency index that may be found using the Revised-RPW method. The CPU times are very close for using each method.

3.3 Comparisons of the methods based on the indicators

Since the proposed GES algorithm with the Revised-RPW method could dominate the GES algorithm with RPW method, the results of this method have been considered to be compared with the related results in the literature. Therefore, the Revised-RPW method as a source for generation of the initial solution for GES algorithm is used. In Table 9, line efficiency (LE which is in percent), variation (VI), and smoothness index (SI), as their formulations presented in Sect. 2, are computed for each of the problem instances using the considered solution methods. The reported results in Table 9, are the average of 10 times execution of this algorithm for each test problem.

In Table 10, these results found by the proposed GES method are compared with the methods proposed by Hwang et al. [18]. Since Hwang et al. have not computed the Smoothness index in their study, the other results such as the number of workstations (OBF), line efficiency, and variation are compared with each other in Table 10.

According to Table 10, all the three methods found the same number of workstations for the mentioned problems. The proposed GES method has reached the same or better line efficiency results for 19 out of 20 cases, in comparison with the other methods. Additionally, the variation results of the proposed method in this study are the same or better than the results of the first method proposed by Hwang et al. [18] with fitness function E, and it is the same or better than 19 results of the method with fitness function of EV. Since there are no more results in the study of Hwang et al. to be compared, and there is no case that their method failed to find its optimal solution. Hence we conclude that the proposed GES method performs as good as the Hwang et al. method.

4 Conclusion

In this research, the GES method was adapted for the U-shaped assembly line balancing problem to obtain an efficient and effective line balancing procedure. After some modifications, an improved version of the classical ranked positional weight method which we called Revised-RPW was proposed to generate an initial solution for GES. Besides, three selection mechanisms were introduced to enhance the performance of GES to generate new solutions. To boost the performance of the proposed algorithm, the COMSOAL method was revisited and improved using the critical path to allocate the activities. Outcomes reveal that the proposed GES method is more efficient than both methods presented by Hwang et al. [18] for solving U-shaped assembly line balancing test instances, in the sense that our method provides the same or better results for all of the test problems except one.

For further studies, the solution approach could be investigated in multi-objective U-shaped line balancing problems like considering the minimization of cycle time as the simultaneous objective functions. Furthermore, all the activities can be done in more than one station, which means that some parts of one activity can be done on one workstation and the rest in another. Using goal programming to optimize such a problem or using other metaheuristics like particle swarm optimization, the neural network can also be one of the future studies.

References

1. Abbasi-Pooya A, Husseinzadeh Kashan A (2017) New mathematical models and a hybrid grouping evolution strategy algorithm for optimal helicopter routing and crew pickup and delivery. Comput Ind Eng 112:35–56
2. Agustı L, Salcedo-Sanz S, Jiménez-Fernández S, Carro-Calvo L, Del Ser J, Portilla-Figueras JA (2012) A new grouping genetic algorithm for clustering problems. Expert Syst Appl 39(10):9695–9703
3. Alavidoost M, Zarandi MF, Tarimoradi M, Nemati Y (2017) Modified genetic algorithm for simple straight and U-shaped assembly line balancing with fuzzy processing times. J Intell Manuf 28(2):313–336
4. Arcus AL (1965) A computer method of sequencing operations for assembly lines. Int J Prod Res 4(4):259–277
5. Brimberg J, Mladenović N, Urošević D (2015) Solving the maximally diverse grouping problem by skewed general variable neighborhood search. Inf Sci 295:650–675
6. Falkenauer E, Delchambre A (1992) A genetic algorithm for bin packing and line balancing. In: IEEE international conference on paper presented at the robotics and automation, 1992. Proceedings, 1992
7. Fathi M, Alvarez M, Rodriguez V (2011) A new heuristic approach to solving U-shape assembly line balancing problems type. World Acad Sci Eng Technol 59:413–421
8. Ghadiri Nejad M, Banar M (2018) Emergency response time minimization by incorporating ground and aerial transportation. Ann Optim Theory Pract 1(1):43–57
9. Ghadiri Nejad M, Husseinzadeh Kashan A, Rismanchian F (2013) A new competitive method for solving assembly line balancing problem. In: 1st international conference on new directions in business, management, finance and economics, Famagusta, Cyprus
10. Ghadiri Nejad M, Shavarani SM, Vizvári B, Vatankhah Barenji R (2018) Trade-off between process scheduling and production cost in cyclic flexible robotic cell. Int J Adv Manuf Technol 96(1–4):1081–1091
11. Glonegger M, Reinhart G (2015) Planning of synchronized assembly lines taking into consideration human performance fluctuations. Prod Eng Res Dev 9(2):277–287
12. Grzechca W (2014) Assembly line balancing problem with reduced number of workstations. IFAC Proc Vol 47(3):6180–6185
13. Helgeson W, Birnie D (1961) Assembly line balancing using the ranked positional weight technique. J Ind Eng 12(6):394–398
14. Husseinzadeh Kashan A, Akbari AA, Ostadi B (2015) Grouping evolution strategies: an effective approach for grouping problems. Appl Math Model 39(9):2703–2720
15. Husseinzadeh Kashan A, Jenabi M, Husseinzadeh Kashan M (2009) A new solution approach for grouping problems based on evolution strategies. In: Paper presented at the 2009 international conference of soft computing and pattern recognition
16. Husseinzadeh Kashan A, Keshmiry M, Dahooie JH, Abbasi-Pooya A (2016) A simple yet effective grouping evolutionary strategy (GES) algorithm for scheduling parallel machines. Neural Comput Appl. https://doi.org/10.1007/s00521-016-2789-3
17. Husseinzadeh Kashan A, Rezaee B, Karimiyan S (2013) An efficient approach for unsupervised fuzzy clustering based on grouping evolution strategies. Pattern Recognit 46(5):1240–1254
18. Hwang RK, Katayama H, Gen M (2008) U-shaped assembly line balancing problem with genetic algorithm. Int J Prod Res 46(16):4637–4649
19. Jonnalagedda V, Dabade B (2014) Application of simple genetic algorithm to U-shaped assembly line balancing problem of type II. IFAC Proc Vol 47(3):6168–6173
20. Kucukkoc I, Zhang DZ (2015) Balancing of parallel U-shaped assembly lines. Comput Oper Res 64(Supplement C):233–244
21. Li Z, Kucukkoc I, Nilakantan JM (2017) Comprehensive review and evaluation of heuristics and meta-heuristics for two-sided assembly line balancing problem. Comput Oper Res 84:146–161

22. Li Z, Tang Q, Zhang L (2017) Two-sided assembly line balancing problem of type I: Improvements, a simple algorithm and a comprehensive study. Comput Oper Res 79:78–93
23. Mosallaeipour S, Ghadiri Nejad M, Shavarani SM, Nazerian R (2018) Mobile robot scheduling for cycle time optimization in flow-shop cells, a case study. Prod Eng Res Dev 12(1):83–93
24. Mukund Nilakantan J, Ponnambalam S (2016) Robotic U-shaped assembly line balancing using particle swarm optimization. Eng Optim 48(2):231–252
25. Ogan D, Azizoglu M (2015). A branch and bound method for the line balancing problem in U-shaped assembly lines with equipment requirements. J Manuf Syst 36(Supplement C), 46–54
26. Oksuz MK, Buyukozkan K, Satoglu SI (2017) U-shaped assembly line worker assignment and balancing problem: A mathematical model and two meta-heuristics. Comput Ind Eng 112(Supplement C):246–263
27. Rechenberg I (1973) Evolution strategy: optimization of technical systems by means of biological evolution, vol 104. Fromman-Holzboog, Stuttgart
28. Reinhart G, Werner J, Lange F (2009) Robot based system for the automation of flow assembly lines. Prod Eng Res Dev 3(1):121–126
29. Scholl A, Klein R (1999) ULINO: Optimally balancing U-shaped JIT assembly lines. Int J Prod Res 37(4):721–736
30. Shavarani SM, Ghadiri Nejad M, 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. Int J Adv Manuf Technol 95(9–12):3141–3153
31. Zhou Y, Hao J-K, Duval B (2016) Reinforcement learning based local search for grouping problems: a case study on graph coloring. Expert Syst Appl 64:412–422