A Greedy Randomized Adaptive Search for Solving Chance-Constrained U-Shaped Assembly Line Balancing Problem

Mohammad Zakaraia, Cairo University, Egypt*
Hegazy Zaher, Cairo University, Egypt
Naglaa Ragaa, Cairo University, Egypt

ABSTRACT

This paper discusses the U-shaped assembly line balancing problem in case of stochastic processing time. The problem is formulated using chance-constrained programming, and the greedy randomized adaptive search procedure is used to solve the problem. In order to prove the efficiency of the proposed algorithm, 71 problems taken from well-known benchmarks are solved and compared with the theoretical lower bound, and 13 of them were compared with another approach used to solve the same problem in another paper, which is beam search. The results show that 59 problems are the same as the theoretical aspiration lower bound. In addition, the results of 11 of 13 problems compared with beam search are the same, and the results of two problems are better than beam search. The t-test statistics is applied and showed that there is no significance difference between the proposed algorithm and the theoretical lower bound; thus, the proposed algorithm shows efficiency when compared with the aspired values of the theoretical lower bound.

KEYWORDS

Chance-Constrained Programming, Greedy Randomized Adaptive Search Procedure, Local Search, Meta-Heuristics, Taguchi Method, U-Shaped Assembly Line Balancing Problem

1. INTRODUCTION

The assembly line is of great importance in industry. It simplifies the assembly processes of the products by implementing them in a set of stations instead of having all the work done by a single skilled worker. It reduces the learning aspects, and it guarantees a fixed time for completing the assembled products. The assembly line balancing problem (ALBP) is such a problem that seeks to optimize the assignment of the assembly tasks to achieve some objectives such as minimizing the number of stations or minimizing the cycle time. The simplified assumptions of the problem contain three constraints. The first one is to assign each task in only one station. The second one is to ensure that each station time does not exceed the cycle time. The third one is to confirm that each task is...
assigned after its predecessors. The U-shaped assembly line balancing problem is one of the assembly
line balancing problems that have one of the practical relevance, which is the U-shaped line. In this
type of assembly lines, the precedence constraints are related to both of the predecessors and successors
of the tasks, where any assembly task must be assigned after either its predecessors or its successors.

This paper discusses the U-shaped assembly line balancing problem in case that the processing
times of the tasks are normally distributed random numbers with known means and variances.
Therefore, the cycle time constraints herein are represented as chance-constraints that realized by
minimum probabilities. In order to solve the problem, the chance-constraints are converted to non-
linear deterministic constraints. The proposed algorithm for solving the problem is one of the single-
solution based metaheuristics, which is greedy randomized adaptive search procedure (GRASP). In
order to have the best results of the proposed algorithm, its parameters are optimized using Taguchi
method. The computational results are constructed by implementing the proposed algorithm on 48
adapted problems selected from well-known deterministic benchmarks found in https://assembly-
line-balancing.de/. The adaptation is done by considering the processing times of the selected
benchmark problems as the expected processing times of the tasks and the variances are calculated
using the method of Carraway (1989). In such method, the variance for each task is to be generated
randomly from \( \left[ \left( \frac{\mu_i}{4} \right)^2 \right] \) for low variances and from \( \left[ \left( \frac{\mu_i}{2} \right)^2 \right] \) for high variances. The selected
problems of for computational results are solved in case that the chance probabilities are equal to
0.90, 0.95, and 0.975. To proof the efficiency of the proposed GRASP algorithm, the computational
results is compared with the results of constrained programming approach found in (Pınarbaşı, 2021).

The paper is organized as follows: the second section presents a literature review of the U-shaped
assembly line balancing problem. The third section shows the methodology used in the paper. The
fourth section presents the proposed mathematical model for CUALBP-1. The fifth section proposes
a GRASP approach to solve the problem. The sixth section produces an experimental design to
optimize the parameter levels of the proposed algorithm. Eventually, the seventh section shows the
computational results.

2. LITERATURE REVIEW

This section shows some information about the previous work in the U-shaped assembly line balancing
problem. The problem was first presented by Miltenburg and Wijngaard (1994), where they developed
dynamic programming approach based on a heuristic to solve the problem. Ajenblit and Wainwright
(1998) developed a genetic algorithm to minimize the number of stations. Nakade and Ohno (1999)
solved the problem by using a heuristic approach. Their paper handled multi objectives, where
they minimized the cycle time and the number of assigned workers. Erel et al. (2001) developed
a simulated annealing approach to minimize the number stations of the problem. Gökçen et al.
(2005) solved the problem by using the shortest route approach, where they sought to minimize the
number of stations. Gökçen and Ağpak (2006) used a multi-criteria decision-making approach for
achieving several conflicting goals.

Baykasoğlu (2006) developed a simulated annealing approach to maximize the smoothness index
and to minimize the number of stations. Kim et al. (2006) presented an endosymbiotic evolutionary
algorithm to maximize workload smoothness. Hwang et al. (2008) developed a genetic algorithm
to optimize multiple objectives, which are minimizing the number of stations and minimizing the
variation of workload. Kara et al. (2009) produced a fuzzy goal programming for U-lines. Özcan
and Toklu (2009) used Simulated annealing and genetic algorithm to maximize the line efficiency
and to maximize the smoothness index. R. Hwang and Katayama (2009) developed an evolutionary
algorithm to minimize the number of stations and the variation of workload. Bagher et al. (2011)
used a hybrid evolutionary algorithm to solve the problem under uncertainty to minimize the number
of stations, total idle time, and non-completion probabilities of each station. Kazemi et al. (2011) developed a genetic algorithm to minimize the total costs associated with the number of stations and task duplication. Hamzadayi and Yildiz (2012) presented a genetic algorithm to minimize the number of stations and maximize the smoothness index. Rabbani et al. (2012) developed a heuristic algorithm based on a genetic algorithm to minimize the number of stations and the cycle time. Avikal et al. (2013) used the critical path method to minimize the number of stations and maximize labor productivity. Hamzadayi and Yildiz (2013) proposed a simulated annealing approach for minimizing the number of stations for mixed models u-shaped assembly line balancing problem.

Fattahi et al. (2014) presented a new formulation for the problem, where the objective is to minimize the number of stations. Jayaswal and Agarwal (2014) developed a simulated annealing approach to minimize the total annual cost of work station utilization, equipment operation, and assistant employment. Hazir and Dolgui (2015) proposed a bender decomposition algorithm for minimizing the cycle time. Ogan and Azizoglu (2015) developed a branch and bound algorithm for minimizing the total equipment cost. Alavidoost et al. (2016) used a two-phase interactive fuzzy programming approach to minimize the number of stations and cycle time under uncertainty. Nilakantan and Ponnambalam (2016) developed a particle swarm optimization approach for minimizing the cycle time and maximizing the production rate. Alavidoost et al. (2017) developed a modified genetic algorithm that deals with the problem under uncertainty. The objectives were to minimize the number of stations, maximize the fuzzy balance efficiency, and minimize the fuzzy idle time percentage.

Li et al. (2017) developed a heuristic approach based on multiple rules to minimize the cycle time. Oksuz et al. (2017) used an artificial bee colony algorithm and genetic algorithm to maximize line efficiency. Z. Li et al. (2018) proposed branch, bound and remember algorithm to maximize the number of stations. Zhang et al. (2019) developed a migrating birds optimization algorithm for minimizing the cycle time. Aydoğan et al. (2019) developed a particle swarm optimization algorithm to minimize the number of stations under uncertainty. Pınarbaşı (2021) dealt with the same problem discussed in this paper through linearizing the non-linear constraints and he used IBM ILOG CP solver to solve them.

The literature review shows that most of the researches have developed approaches to deal with the problem in its deterministic case. Hence, it appears that it is a solid motivation to discuss the uncertainty of the problem further. Therefore, this paper presents one of the uncertainty cases of the problem, which is the chance-constrained problem. In addition, it proposes an efficient GRASP algorithm as a new approach for solving the problem, where its results are compared with the results found in (Pınarbaşı, 2021) and proved a high efficiency in terms of objective values and CPU times.

3. PROBLEM FORMULATION

According to the literature review, it appears that most researches have focused on deterministic processing time. In this paper, processing times of the tasks are represented as random variables that are normally distributed and each has an expected value $E(t_i)$ and variance $Var(t_i)$. The cycle time constraints of the problem are realized with a minimum probability. So, the problem can be reformulated as shown in Table 1.

\[
\text{Minimize } \sum_{j=1}^{n} y_j 
\]

(1)

\[
s.t. \sum_{j=1}^{n} x_{ij} = 1, \forall i = \{1, 2, \ldots, n\} 
\]

(2)
Table 1. Notations

| Symbol | Description |
|--------|-------------|
| $i = \{1, 2, \ldots, n\}$ | The set of tasks |
| $j = \{1, 2, \ldots, n\}$ | The set of stations |
| $t_i$ | The processing time |
| $ct$ | The cycle time |
| $IP(i)$ | The set of immediate predecessors of task $i$ |
| $IS(i)$ | The set of immediate successors of task $i$ |

**Equations**

1. $x_{ij} = \begin{cases} 1, & \text{if task } i \text{ is assigned to station } j \\ 0, & \text{otherwise} \end{cases}$
2. $y_j = \begin{cases} 1, & \text{if } \sum_{i=1}^{n} x_{ij} \geq 1 \\ 0, & \text{otherwise} \end{cases}$
3. $P_i = \begin{cases} 1, & \text{if } \sum_{j=1}^{n} x_{kj} \leq \sum_{j=1}^{n} x_{ij}, \forall k_i \in IP(i) \\ 0, & \text{otherwise} \end{cases}$
4. $S_i = \begin{cases} 1, & \text{if } \sum_{j=1}^{n} x_{kj} \leq \sum_{j=1}^{n} x_{ij}, \forall k_i \in IS(i) \\ 0, & \text{otherwise} \end{cases}$

$$P \left( \sum_{i=1}^{n} x_{ij} \leq y_j \cdot ct \right) \geq \alpha_j, \forall j = \{1, 2, \ldots, n\}$$ (3)

$$P_i + S_i \geq 1, \forall i = \{1, 2, \ldots, n\}$$ (4)

$$x_{ij} \in \{0, 1\}, \forall i, j$$ (5)
The objective function (1) seeks to minimize the number of stations. The set of constraints (2) represents the assignment constraints, which ensures that each task is assigned in only one station. The set of chance constraints (3) ensures that the total processing time of any station doesn’t exceed the cycle time with probability greater than or equal to \(\alpha_j\). Such set of chance constraints can be converted to the following set of constraints to overcome the probabilities (Taha, 2017):

\[
\sum_{i=1}^{n} E(t_{ij}) x_{ij} + k_{\alpha_j} \sqrt{\sum_{j} \text{var}(t_{ij}) x_{ij}} \leq y_{ct}, \forall j = \{1, 2, \ldots, n\}, \text{where } k_{\alpha_j} \text{ is the z-score of } \alpha_j
\]

The set of constraints (4) shows that each task has to be assigned after assigning either its immediate predecessors, or its immediate successors. Eventually, the set of constraints (5) ensures that the problem is an integer 0-1 programming problem.

4. THE PROPOSED ALGORITHM

The proposed approach for solving the problem is the greedy randomized adaptive search procedure. This algorithm is one of the meta-heuristics that searches for optimized solutions, through a set of iterations that contains a solution construction and a local search. The solution construction of this approach begins with a stochastic greedy rule that leads to a candidate solution. The next step of GRASP is to search around the candidate solution by using a local search. The final step is to compare the evaluation of the best local solution with the evaluation of the best solution found per all previous iterations.

4.1. Algorithm 1 (The Solution Construction)

The proposed approach begins with the solution structure, which can be developed as seen in algorithm 1. Figure 1 shows the flowchart of algorithm 1.

Algorithm 1 illustrates the random solution construction of the proposed algorithm. It begins by constructing a random arrangement (RA) of the tasks. The iterations of the algorithm find the assignable tasks according to the problem constraints, and then they choose the top task in the constructed random arrangement and assign it to the current open station. The assigned task then

Algorithm 1. Solution construction

| Input: RA, Random arrangement of the problem tasks |
| Station, \(S = 1\) |
| Initiate Solution = \(\varnothing\) |
| While RA is not empty DO |
| Assignable tasks = The set of tasks that confirms constraints (2), (3), (4), and (5) |
| If Assignable tasks = \(\varnothing\) and RA = \(\varnothing\) then set \(S = S + 1\) |
| If Assignable tasks = \(\varnothing\) and RA = \(\varnothing\) then Break While loop |
| AT = The first task in RA that belongs to Assignable tasks |
| Solution = Solution \(\cup\) (AT, S) |
| Remove AT from RA |
| End While |
| Return Solution |
will be eliminated from the constructed random arrangement set. A new station in the algorithm is
to be opened when there are existed tasks in the constructed random arrangement set and no more
of them can fit in the current open station. The solution of the algorithm will be returned when all
tasks are assigned. Figure 1 shows the flowchart of algorithm 1.

4.2. Algorithm 2 (The Local Search)

According to the concept of GRASP, the constructed solution should be followed by a local search
that tries to find the optimal local neighbor of it. The proposed mutation technique to find a neighbor
is to swap the tasks of any station either with a forward station or backward station. After swapping,
the sequence of the tasks will be presented to Algorithm 1 as RA to obtain another solution that is
neighbor to the main random constructed solution. Algorithm 2 shows the local search of the proposed
GRASP algorithm for solving the problem. Figure 2 shows the flowchart of the algorithm 2.

4.3. Algorithm 3 (GRASP)

After illustrating the constructed solution procedures, and the local search procedures, the steps of
the proposed GRASP algorithm can be illustrated in Algorithm 3. Figure 3 shows the flowchart of
algorithm 3.

5. EXPERIMENTAL DESIGN

The proposed algorithm is coded using python in a PC that has core 2 due CPU with 2.93 GHz and
4 GB rams. The proposed algorithm has four parameters, which are the number of constructed
Algorithm 2. Local search

Input: Solution
Set Best Solution = Solution
Max = Number of iterations
i = 1
While i <= Max then
    Swap the tasks of any two random stations of Solution and save the output sequence of tasks in Swapped Tasks
    Run Algorithm 1 by setting RA = Swapped Tasks and return the output as Solution2
    If f(Solution 2) < f(Solution) then Best solution = Solution2
    i = i + 1
Return Best solution

Figure 2. Algorithm 2 “Local Search”

solutions (C), the number of local solutions (L), the swapping rate (S), and mutation function (M). The levels of parameter C are 5, 15, 20 solutions. The levels of parameter L are 10, 20, and 30. The levels of parameter S are 0.2, 0.3, and 0.4, where in such parameter the number of swapped tasks is obtained by multiplying the swapping rate by the number of tasks and rounding up the output to the
Algorithm 3. GRASP for solving CUALBP-1

Input: Max
i = 1
Global Best = ∅
while i ≤ Max then
    Solution = Algorithm 1
    Best Local Solution = Algorithm 2(Solution)
    If f(Best Local Evaluation) < f(Global Best Evaluation) then
        Global Best = Best Local Solution
        i = i + 1
    Return Global Best

Figure 3. Algorithm 3 “GRASP for Solving CUALBP-1”
nearest integer. The levels of parameter M are forward swapping (FS), backward swapping (BS), and bi-directional swapping (DS). The required trails to make full factorial design are $4^3 = 64$ trails. In Taguchi method, such number of trails can be reduced after calculating the degrees of freedom for each parameter to select the right orthogonal array. Equation (7) shows the required number of trails $N_{Taguchi}$, where $L_i$ represents level $i$ and $NV$ is the number of levels:

$$N_{Taguchi} = 1 + \sum_{i=1}^{NV} (L_i - 1)$$

(7)

So, the required number of trails should be greater than or equal 9. Therefore, the selected orthogonal array for this experiment is $L_9$, which is in Table 3.

The response in this experimental design includes the CPU time beside the objective function value as seen in equation (8). The selected test optimization problems are taken from a well-known benchmark found in https://assembly-line-balancing.de/. Table 4 shows the selected optimization problems and their associated cycle times and sizes. Table 5 shows the normalized results for each trail in each problem. The normalization of the results is done due to the variation of the objective functions from problem to another:

$$Response = \frac{\sum_{j=1}^{n} y_j - 1}{CPU\ time}$$

(8)

Table 2. Notations

| M     | Mutation function |
|-------|-------------------|
| S     | The number of swapped stations |
| L     | The number of local search solutions |
| C     | The number of constructed solutions |

Table 3. The selected orthogonal array for experimental design

| M     | S   | L  | C  |
|-------|-----|----|----|
| FS    | 0.2 | 10 | 5  |
| BS    | 0.3 | 20 | 5  |
| DS    | 0.4 | 30 | 5  |
| DS    | 0.3 | 10 | 15 |
| FS    | 0.4 | 20 | 15 |
| BS    | 0.2 | 30 | 15 |
| BS    | 0.4 | 10 | 20 |
| DS    | 0.2 | 20 | 20 |
| FS    | 0.3 | 30 | 20 |
In order to obtain a robust setting that minimizes the response with a higher accuracy, the main effects for means and standard deviations are studied. Figure 4 shows the main effects plot for means and Figure 5 shows the main effects plot for standard deviations. From the figures, it can be stated that the optimized parameter levels can be summarized in Table 6, where each selected level has the minimum response and standard deviation.

### 6. COMPUTATIONAL RESULTS

The computational results show the implementation of the proposed GRASP algorithm in 48 adapted problems taken from https://assembly-line-balancing.de/. The problems vary in their sizes from 7 to 148 tasks. The comparison is done with a constrained programming approach (CP) proposed by Pınarbaşı (2021). The problems in the benchmark are deterministic. Therefore, the adaptation is done by considering the expected processing times of the tasks are the same as the original processing times in benchmark and the variances are calculated using a modified version of the method of Carraway (1989), which is mentioned in the introduction section. The reason for modifying the method is that it found that sometimes the generated variance causes violating the cycle time constraints. Therefore, the modified version of the method of Carraway (1989) is shown in equation 10, where it based on the generated value $Var_{\text{test}}\left(t_i\right)$ from either the low variances interval $\left[0,\left(E\left(t_i\right) / 4\right)^2\right]$ or the high variances interval $\left[0,\left(E\left(t_i\right) / 2\right)^2\right]$. The computational results table has a column for

| Problem   | Cycle time | Size |
|-----------|------------|------|
| JACKSON   | 9          | 11   |
| MITCHELL  | 14         | 21   |
| HESKIA    | 138        | 28   |
| SAWYER30  | 25         | 30   |
| ARC83     | 5048       | 83   |
| ARC111    | 5755       | 111  |

| Problem   | Cycle time | Size |
|-----------|------------|------|
| JACKSON   | 9          | 11   |
| MITCHELL  | 14         | 21   |
| HESKIA    | 138        | 28   |
| SAWYER30  | 25         | 30   |
| ARC83     | 5048       | 83   |
| ARC111    | 5755       | 111  |

Table 4. Test optimization problems

| Problem   | Cycle time | Size |
|-----------|------------|------|
| JACKSON   | 9          | 11   |
| MITCHELL  | 14         | 21   |
| HESKIA    | 138        | 28   |
| SAWYER30  | 25         | 30   |
| ARC83     | 5048       | 83   |
| ARC111    | 5755       | 111  |

Table 5. The normalized results for each trail

| JACKSON   | MITCHELL | HESKIA | SAWYER30 | ARC83 | ARC111 |
|-----------|----------|--------|----------|-------|--------|
| 0.1269    | 0.1110   | 0.1224 | 0.1138   | 0.1166| 0.1140 |
| 0.1089    | 0.1111   | 0.1226 | 0.1139   | 0.1099| 0.1107 |
| 0.1092    | 0.1112   | 0.1076 | 0.1140   | 0.1105| 0.1118 |
| 0.1091    | 0.1110   | 0.1076 | 0.1138   | 0.1094| 0.1109 |
| 0.1093    | 0.1111   | 0.1076 | 0.1138   | 0.1094| 0.1109 |
| 0.1095    | 0.1112   | 0.1084 | 0.1138   | 0.1094| 0.1102 |
| 0.1090    | 0.1110   | 0.1090 | 0.1138   | 0.1094| 0.1102 |
| 0.1089    | 0.1111   | 0.1077 | 0.1080   | 0.1107| 0.1114 |
| 0.1090    | 0.1112   | 0.1075 | 0.1078   | 0.1113| 0.1104 |
Figure 4. Mean effects plot for means

Figure 5. Mean effects plot for standard deviations
International Journal of Applied Metaheuristic Computing
Volume 13 • Issue 1

12

the deterministic lower bound for each problem to show how far the uncertainty results from the deterministic lower bound, which can be found by equation (11). The problems are solved in case of chance probabilities that are equal to 0.90, 0.95, and 0.975. Table 7 shows the computational results in case of low variances and Table 8 shows the computational results in case of high variances. Note: all CPU times in Tables 7 and 8 are in seconds:

\[ \text{Var}_{test}(t_i) = \text{rand}(0,1) \times \left( \frac{E(t_i)}{Var_{size}} \right)^2, \text{Var}_{size} = 2 \text{ or } 4 \]  (9)

\[ \text{Var}(t_i) = \begin{cases} \text{Var}_{test}(t_i), & \text{if } \text{Var}_{test}(t_i) \leq ct \\ \left( \frac{ct - E(t_i)}{1.96} \right)^2, & \text{Otherwise} \end{cases} \]  (10)

\[ LB = \frac{\sum_{i=1}^{n} E(t_i)}{ct}, \text{where } x \text{ is the largest integer greater than } x \]  (11)

The proposed GRASP algorithm proves a high efficiency in terms of objective values and CPU times when compared to CP approach (Pınarbaşı, 2021). In case of low variances, the objective values show that the GRASP algorithm is better than CP approach in 33 problems. In case of high variances, the GRASP algorithm is better than CP approach in 45 problems. In terms of CPU times, Pınarbaşı (2021) reported in his research that the minimum CPU time for the small sized problems, which are lower than or equal 70 tasks, is 0.01 and the maximum CPU time is 300. The large sized problems in CP approach have minimum CPU time approximately equal to the maximum CPU time, which both equal to 900. While in GRASP algorithm, the minimum CPU time is 0.01 and the maximum CPU time is 1.69 for all problems.

7. CONCLUSION

This paper presents a GRASP algorithm as a new approach for solving the U-shaped assembly line balancing problem under uncertainty. The uncertainty herein is related to having the processing times of the tasks as normally distributed random variables with known means and variances. Therefore, the problem is formulated as chance-constrained programming by considering that the cycle time

| Parameter | Level |
|-----------|-------|
| M         | DS    |
| S         | 0.4   |
| L         | 30    |
| C         | 15    |

Table 6. The optimized parameters
| Problem    | ct   | LB   | \( \alpha = 0.90 \) | \( \alpha = 0.95 \) | \( \alpha = 0.975 \) |
|------------|------|------|----------------------|----------------------|----------------------|
|            | CP   | Number of stations (GRASP) | CPU time (GRASP) | CP   | Number of stations (GRASP) | CPU time (GRASP) | CP   | Number of stations (GRASP) | CPU time (GRASP) |
| B          | 8    | 4    | 5                    | 0.01                | 5      | 0.01                | 5      | 0.01 |
| MERTENS    | 10   | 3    | 4                    | 0.01                | 4      | 0.00                | 4      | 0.01 |
| MERTENS    | 15   | 2    | 3                    | 0.01                | 3      | 0.00                | 3      | 0.00 |
| MERTENS    | 18   | 2    | 2                    | 0.01                | 2      | 0.00                | 2      | 0.00 |
| BOWMAN8    | 20   | 4    | 6                    | 0.01                | 6      | 0.01                | 5      | 0.01 |
| JAESCHKE   | 6    | 7    | 8                    | 0.01                | 8      | 0.01                | 8      | 0.01 |
| JAESCHKE   | 7    | 6    | 7                    | 0.01                | 7      | 0.01                | 7      | 0.01 |
| JAESCHKE   | 8    | 5    | 7                    | 0.01                | 7      | 0.01                | 7      | 0.01 |
| JAESCHKE   | 10   | 4    | 5                    | 0.01                | 5      | 0.01                | 5      | 0.01 |
| JAESCHKE   | 18   | 3    | 3                    | 0.01                | 3      | 0.02                | 3      | 0.01 |
| JACKSON    | 9    | 6    | 7                    | 0.03                | 7      | 0.03                | 7      | 0.01 |
| JACKSON    | 10   | 5    | 7                    | 0.01                | 7      | 0.01                | 7      | 0.01 |
| JACKSON    | 13   | 4    | 5                    | 0.01                | 5      | 0.01                | 5      | 0.01 |
| JACKSON    | 14   | 4    | 4                    | 0.01                | 4      | 0.01                | 4      | 0.01 |
| JACKSON    | 21   | 3    | 3                    | 0.01                | 3      | 0.01                | 3      | 0.01 |
| MITCHELL   | 15   | 7    | NFS                  | 0.03                | NFS    | 0.03                | 9      | 0.03 |
| MITCHELL   | 21   | 5    | 6                    | 0.02                | 6      | 0.02                | 6      | 0.03 |
| MITCHELL   | 26   | 5    | 5                    | 0.02                | 5      | 0.02                | 5      | 0.02 |
| MITCHELL   | 35   | 3    | 4                    | 0.25                | 4      | 0.07                | 4      | 0.14 |
| MITCHELL   | 39   | 3    | 4                    | 0.02                | 4      | 0.02                | 3      | 0.02 |
| HESKIA     | 205  | 5    | 6                    | 0.06                | 6      | 0.06                | 6      | 0.06 |
| HESKIA     | 216  | 5    | 6                    | 0.11                | 6      | 0.05                | 6      | 0.11 |
| HESKIA     | 256  | 4    | 5                    | 0.06                | 5      | 0.05                | 5      | 0.33 |
| HESKIA     | 324  | 4    | 4                    | 0.05                | 4      | 0.05                | 4      | 0.05 |
| HESKIA     | 342  | 3    | 4                    | 0.05                | 4      | 0.05                | 4      | 0.33 |
| SAWYER30   | 33   | 10   | 15                   | 0.11                | 14     | 0.19                | 14     | 0.06 |
| SAWYER30   | 41   | 8    | 11                   | 0.05                | 11     | 0.05                | 10     | 0.05 |
| SAWYER30   | 47   | 7    | 10                   | 0.05                | 9      | 0.05                | 8      | 0.05 |
| KILBRID    | 79   | 7    | 9                    | 0.11                | 8      | 0.11                | 8      | 0.11 |
| KILBRID    | 92   | 6    | 7                    | 0.10                | 7      | 0.11                | 7      | 0.11 |
| KILBRID    | 110  | 6    | 6                    | 0.12                | 6      | 0.11                | 6      | 0.11 |
| KILBRID    | 138  | 4    | 5                    | 0.83                | 5      | 0.10                | 5      | 0.11 |
| KILBRID    | 184  | 3    | 4                    | 0.52                | 4      | 0.10                | 4      | 0.21 |
| TONGE70    | 207  | 17   | 22                   | 0.61                | 20     | 2.79                | 20     | 3.41 |
| TONGE70    | 234  | 15   | 19                   | 0.29                | 19     | 0.63                | 18     | 0.30 |
| TONGE70    | 320  | 11   | 14                   | 0.27                | 13     | 0.28                | 13     | 0.28 |

continued on following page
Table 7. Continued

| Problem  | ct | LB | α = 0.90 | α = 0.95 | α = 0.975 |
|----------|----|----|----------|----------|-----------|
|          | CP | Number of stations (GRASP) | CPU time (GRASP) | CP | Number of stations (GRASP) | CPU time (GRASP) | CP | Number of stations (GRASP) | CPU time (GRASP) |
| LUTZ2    | 18 | 27 | 36 | 31 | 0.49 | 34 | 31 | 0.50 | 33 | 31 | 1.96 |
| LUTZ2    | 21 | 24 | 30 | 26 | 0.45 | 29 | 26 | 0.46 | 28 | 26 | 0.92 |
| LUTZ2    | 32 | 16 | 20 | 16 | 3.59 | 19 | 17 | 0.41 | 18 | 16 | 2.03 |
| MUKHERJE | 351 | 12 | 15 | 13 | 0.60 | 15 | 13 | 0.61 | 14 | 13 | 0.61 |
| MUKHERJE | 471 | 9 | 11 | 10 | 0.58 | 11 | 10 | 0.57 | 11 | 10 | 0.60 |
| MUKHERJE | 704 | 6 | 8 | 7 | 0.57 | 8 | 6 | 6.20 | 7 | 7 | 0.59 |
| ARC111   | 9554 | 16 | 20 | 17 | 0.82 | 19 | 17 | 1.63 | 18 | 17 | 5.03 |
| ARC111   | 11570 | 13 | 16 | 14 | 0.87 | 16 | 14 | 1.61 | 15 | 14 | 1.64 |
| ARC111   | 15040 | 10 | 13 | 11 | 0.78 | 12 | 11 | 0.78 | 12 | 11 | 0.80 |
| BARTHOLD | 470 | 12 | 15 | 13 | 1.65 | 15 | 13 | 1.58 | 14 | 13 | 1.63 |
| BARTHOLD | 795 | 8 | 9 | 8 | 1.63 | 9 | 8 | 1.65 | 9 | 8 | 1.63 |
| BARTHOLD | 1127 | 5 | 7 | 6 | 1.71 | 6 | 6 | 1.67 | 6 | 6 | 1.67 |

Table 8. Computational results (high variances)

| Problem  | ct | LB | α = 0.90 | α = 0.95 | α = 0.975 |
|----------|----|----|----------|----------|-----------|
|          | CP | Number of stations (GRASP) | CPU time (GRASP) | CP | Number of stations (GRASP) | CPU time (GRASP) | CP | Number of stations (GRASP) | CPU time (GRASP) |
| MERTENS  | 8  | 4 | NFS 5 | 0.01 | NFS 5 | 0.01 | 5  | 5 | 0.01 |
| MERTENS  | 10 | 3 | 5 | 3 | 0.01 | 5  | 3 | 0.01 | 5  | 4 | 0.01 |
| MERTENS  | 15 | 2 | 3 | 2 | 0.00 | 3  | 2 | 0.00 | 3  | 2 | 0.01 |
| MERTENS  | 18 | 2 | 3 | 2 | 0.00 | 3  | 2 | 0.00 | 2  | 2 | 0.00 |
| BOWMAN8  | 20 | 4 | 6 | 5 | 0.01 | 6  | 5 | 0.01 | 6  | 5 | 0.01 |
| JAIESCHE  | 6 | 7 | NFS 8 | 0.01 | NFS 8 | 0.01 | NFS 8 | 0.01 |
| JAIESCHE  | 7 | 6 | NFS 7 | 0.01 | NFS 7 | 0.03 | NFS 7 | 0.01 |
| JAIESCHE  | 8 | 5 | 7 | 6 | 0.01 | 7  | 6 | 0.01 | 7  | 7 | 0.01 |
| JAIESCHE  | 10 | 4 | 7 | 5 | 0.01 | 7  | 5 | 0.01 | 7  | 5 | 0.01 |
| JAIESCHE  | 18 | 3 | 3 | 3 | 0.01 | 3  | 3 | 0.01 | 3  | 3 | 0.01 |
| JACKSON  | 9 | 6 | NFS 6 | 0.02 | NFS 6 | 0.01 | NFS 6 | 0.01 |
| JACKSON  | 10 | 5 | NFS 5 | 0.01 | NFS 5 | 0.01 | 6  | 5 | 0.01 |
| JACKSON  | 13 | 4 | 5 | 4 | 0.01 | 5  | 4 | 0.01 | 5  | 4 | 0.01 |
| JACKSON  | 14 | 4 | 5 | 4 | 0.01 | 5  | 4 | 0.01 | 5  | 4 | 0.01 |
| JACKSON  | 21 | 3 | 3 | 3 | 0.01 | 3  | 3 | 0.01 | 3  | 3 | 0.01 |
| MITCHELL | 15 | 7 | NFS 8 | 0.03 | NFS 8 | 0.03 | NFS 8 | 0.03 |
| MITCHELL | 21 | 5 | 8 | 6 | 0.03 | 7  | 5 | 0.10 | 7  | 6 | 0.02 |

continued on following page
After optimizing the parameter of the algorithm using Taguchi method, the computational results are done to compare the proposed algorithm with CP approach proposed by Pınarbaşı (2021). The results of the proposed GRASP algorithm show high efficiency in terms of objective values and CPU times.

The future points of research may consider the following points:

### Table 8. Continued

| Problem   | ct  | LB | α = 0.90                |                      | α = 0.95                |                      | α = 0.975                |                      |
|-----------|-----|----|-------------------------|----------------------|-------------------------|----------------------|-------------------------|----------------------|
|           | CP  |     | Number of stations (GRASP) | CPU time (GRASP)     | Number of stations (GRASP) | CPU time (GRASP)     | Number of stations (GRASP) | CPU time (GRASP)     |
| MITCHELL  | 26  | 5  | 5                       | 0.02                 | 6                       | 0.02                 | 5                       | 0.02                 |
| MITCHELL  | 35  | 3  | 4                       | 0.14                 | 4                       | 0.02                 | 4                       | 0.02                 |
| MITCHELL  | 39  | 3  | NFS                     | 0.02                 | NFS                     | 0.03                 | NFS                     | 0.02                 |
| HESKIA    | 205 | 5  | 6                       | 0.06                 | 7                       | 0.06                 | 7                       | 0.06                 |
| HESKIA    | 216 | 5  | 7                       | 0.06                 | 7                       | 0.05                 | 6                       | 0.17                 |
| HESKIA    | 256 | 4  | 6                       | 0.05                 | 6                       | 0.09                 | 5                       | 0.05                 |
| HESKIA    | 324 | 4  | 5                       | 0.05                 | 4                       | 0.05                 | 4                       | 0.06                 |
| HESKIA    | 342 | 3  | 4                       | 0.33                 | 4                       | 0.31                 | 4                       | 0.05                 |
| SAWYER30  | 33  | 10 | NFS                     | 0.06                 | NFS                     | 0.05                 | NFS                     | 0.11                 |
| SAWYER30  | 41  | 8  | 9                       | 0.05                 | 13                      | 0.05                 | 11                      | 0.05                 |
| SAWYER30  | 47  | 7  | 8                       | 0.05                 | 11                      | 0.05                 | 10                      | 0.05                 |
| KILBRID   | 79  | 7  | NFS                     | 0.11                 | NFS                     | 0.11                 | NFS                     | 0.11                 |
| KILBRID   | 92  | 6  | 7                       | 0.11                 | 8                       | 0.11                 | 8                       | 0.11                 |
| KILBRID   | 110 | 6  | 7                       | 0.11                 | 6                       | 0.11                 | 6                       | 0.10                 |
| KILBRID   | 138 | 4  | 6                       | 0.10                 | 6                       | 0.11                 | 5                       | 0.10                 |
| KILBRID   | 184 | 3  | 4                       | 0.10                 | 4                       | 0.32                 | 4                       | 0.11                 |
| TONGE70   | 207 | 17 | NFS                     | 4.01                 | NFS                     | 1.54                 | NFS                     | 0.30                 |
| TONGE70   | 234 | 15 | NFS                     | 0.60                 | NFS                     | 0.60                 | NFS                     | 0.29                 |
| TONGE70   | 320 | 11 | 16                      | 0.28                 | 15                      | 0.28                 | 14                      | 0.28                 |
| LUTZ2     | 18  | 27 | 40                      | 0.49                 | 39                      | 3.48                 | 35                      | 0.46                 |
| LUTZ2     | 21  | 24 | 32                      | 2.30                 | 32                      | 5.49                 | 29                      | 2.71                 |
| LUTZ2     | 32  | 16 | 24                      | 0.41                 | 23                      | 0.41                 | 20                      | 0.41                 |
| MUKHERJE  | 351 | 12 | 18                      | 0.62                 | 17                      | 0.61                 | 16                      | 0.61                 |
| MUKHERJE  | 471 | 9  | 13                      | 0.60                 | 13                      | 0.59                 | 12                      | 0.60                 |
| MUKHERJE  | 704 | 6  | 9                       | 0.59                 | 9                       | 0.59                 | 8                       | 0.58                 |
| ARC111    | 9554| 16 | 24                      | 4.20                 | 22                      | 1.69                 | 21                      | 2.53                 |
| ARC111    | 11570| 13 | 20                      | 0.81                 | 19                      | 1.68                 | 17                      | 0.81                 |
| ARC111    | 15040| 10 | 15                      | 0.79                 | 15                      | 0.79                 | 13                      | 0.79                 |
| BARTHOLD  | 470 | 12 | 18                      | 1.64                 | 17                      | 1.65                 | 16                      | 1.65                 |
| BARTHOLD  | 795 | 8  | 11                      | 1.62                 | 10                      | 1.63                 | 10                      | 1.63                 |
| BARTHOLD  | 1127| 3  | 8                       | 1.65                 | 7                       | 1.67                 | 7                       | 1.69                 |
• Solving other types of assembly line balancing problems under uncertainty using the same algorithm.
• Adding space constraints that are related to having area for each task.
• Considering that each station has chance probability differs from the other stations, since there are operators more skillful than the others and they should have different chance probability.
• Solving the same problem in case that the number of stations is given and the cycle time is required to be minimized.
• Solving the problem in case that both of the number of stations and the cycle time are not given and the required objective is to maximize the line efficiency.

FUNDING AGENCY

The publisher has waived the Open Access Processing fee for this article.
REFERENCES

Ajenblit, D. A., & Wainwright, R. L. (1998). Applying genetic algorithms to the U-shaped assembly line balancing problem. *Proceedings of the IEEE Conference on Evolutionary Computation, ICEC*, 96–101. doi:10.1109/ICEC.1998.699329

Alavidoost, M. H., Babazadeh, H., & Sayyari, S. T. (2016). An interactive fuzzy programming approach for bi-objective straight and U-shaped assembly line balancing problem. *Applied Soft Computing, 40*, 221–235. doi:10.1016/j.asoc.2015.11.025

Alavidoost, M. H., Zarandi, M. H. F., Tarimoradi, M., & Nemati, Y. (2017). Modified genetic algorithm for simple straight and U-shaped assembly line balancing with fuzzy processing times. *Journal of Intelligent Manufacturing, 28*(2), 313–336. doi:10.1007/s10845-014-0978-4

Avikal, S., Jain, R., Mishra, P. K., & Yadav, H. C. (2013). A heuristic approach for U-shaped assembly line balancing to improve labor productivity. *Computers & Industrial Engineering, 64*(4), 895–901. doi:10.1016/j.cie.2013.01.001

Aydoğan, E. K., Delice, Y., Özcan, U., Gencer, C., & Bali, Ö. (2019). Balancing stochastic U-lines using particle swarm optimization. *Journal of Intelligent Manufacturing, 30*(1), 97–111. doi:10.1007/s10845-016-1234-x

Bagher, M., Zandieh, M., & Farsijiani, H. (2011). Balancing of stochastic U-type assembly lines: An imperialist competitive algorithm. *International Journal of Advanced Manufacturing Technology, 54*(1–4), 271–285. doi:10.1007/s00170-010-2937-3

Baykasoğlu, A. (2006). Multi-rule multi-objective simulated annealing algorithm for straight and U type assembly line balancing problems. *Journal of Intelligent Manufacturing, 17*(2), 217–232. doi:10.1007/s10845-005-6638-y

Carraway, R. L. (1989). *A Dynamic Programming Approach to Stochastic Assembly Line Balancing*. 10.1287/mnsc.35.4.459

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. doi:10.1080/00207540110051905

Fattahi, A., Elaoud, S., Sadeqi Azer, E., & Turkay, M. (2014). A novel integer programming formulation with logic cuts for the U-shaped assembly line balancing problem. *International Journal of Production Research, 52*(5), 1318–1333. doi:10.1080/00207543.2013.832489

Gökçen, H., & Ağpak, K. (2006). A goal programming approach to simple U-line balancing problem. *European Journal of Operational Research, 171*(2), 577–585. doi:10.1016/j.ejor.2004.09.021

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. doi:10.1016/j.apm.2004.10.003

Hamzadayi, A., & Yildiz, G. (2012). A genetic algorithm based approach for simultaneously balancing and sequencing of mixed-model U-lines with parallel workstations and zoning constraints. *Computers & Industrial Engineering, 62*(1), 206–215. doi:10.1016/j.cie.2011.09.008

Hamzadayi, A., & Yildiz, G. (2013). A simulated annealing algorithm based approach for balancing and sequencing of mixed-model U-lines. *Computers & Industrial Engineering, 66*(4), 1070–1084. doi:10.1016/j.cie.2013.08.008

Hazir, Ö., & Dolgui, A. (2015). A decomposition based solution algorithm for U-type assembly line balancing with interval data. *Computers & Operations Research, 59*, 126–131. doi:10.1016/j.cor.2015.01.010

Hwang, R., & Katayama, H. (2009). A multi-decision genetic approach for workload balancing of mixed-model U-shaped assembly line systems. *International Journal of Production Research, 47*(14), 3797–3822. doi:10.1080/00207540701851772

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. doi:10.1080/00207540701247906

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. doi:10.1016/j.jmsy.2014.05.002
Kara, Y., Paksoy, T., & Ter Chang, C. (2009). Binary fuzzy goal programming approach to single model straight and U-shaped assembly line balancing. *European Journal of Operational Research, 195*(2), 335–347. doi:10.1016/j.ejor.2008.01.003

Kazemi, S. M., Ghodsi, R., Rabbani, M., & Tavakkoli-Moghaddam, R. (2011). A novel two-stage genetic algorithm for a mixed-model U-line balancing problem with duplicated tasks. *International Journal of Advanced Manufacturing Technology, 55*(9–12), 1111–1122. doi:10.1007/s00170-010-3120-6

Kim, Y. K., Kim, J. Y., & Kim, Y. (2006). An endosymbiotic evolutionary algorithm for the integration of balancing and sequencing in mixed-model U-lines. *European Journal of Operational Research, 168*(3), 838–852. doi:10.1016/j.ejor.2004.07.032

Li, M., Tang, Q., Zheng, Q., Xia, X., & Floudas, C. A. (2017). Rules-based heuristic approach for the U-shaped assembly line balancing problem. *Applied Mathematical Modelling, 48*, 423–439. doi:10.1016/j.apm.2016.12.031

Li, Z., Kucukkoc, I., & Zhang, Z. (2018). Branch, bound and remember algorithm for U-shaped assembly line balancing. *Computers & Industrial Engineering, 124*, 24–35. doi:10.1016/j.cie.2018.06.037

Miltonburg, G. J., & Wijngaard, J. (1994). U-line line balancing problem. *Management Science, 40*(10), 1378–1388. doi:10.1287/mnsc.40.10.1378

Mukund Nilakantan, J., & Ponnambalam, S. G. (2016). Robotic U-shaped assembly line balancing using particle swarm optimization. *Engineering Optimization, 48*(2), 231–252. doi:10.1080/0305215X.2014.998664

Nakade, K., & Ohno, K. (1999). Optimal worker allocation problem for a U-shaped production line. *International Journal of Production Economics, 60*, 353–358. doi:10.1016/S0925-5273(98)00145-5

Ogan, D., & Azizoglu, M. (2015). A branch and bound method for the line balancing problem in U-shaped assembly lines with equipment requirements. *Journal of Manufacturing Systems, 36*, 46–54. doi:10.1016/j.jmsy.2015.02.007

Oksuz, M. K., Buyukozkan, K., & Satoglu, S. I. (2017). U-shaped assembly line worker assignment and balancing problem: A mathematical model and two meta-heuristics. *Computers & Industrial Engineering, 112*, 246–263. doi:10.1016/j.cie.2017.08.030

Özcan, U., & Toklu, B. (2009). A new hybrid improvement heuristic approach to simple straight and U-type assembly line balancing problems. *Journal of Intelligent Manufacturing, 20*(1), 123–136. doi:10.1007/s10845-008-0108-2

Pınarbaşı, M. (2021). New chance-constrained models for U-type stochastic assembly line balancing problem. *Soft Computing, 25*(14), 9559–9573. Advance online publication. doi:10.1007/s00500-021-05921-z

Rabbani, M., Moghaddam, M., & Manavizadeh, N. (2012). Balancing of mixed-Model two-Sided assembly lines with multiple u-Shaped layout. *International Journal of Advanced Manufacturing Technology, 59*(9–12), 1191–1210. doi:10.1007/s00170-011-3545-6

Zhang, Z., Tang, Q., Han, D., & Li, Z. (2019). Enhanced migrating birds optimization algorithm for U-shaped assembly line balancing problems with workers assignment. *Neural Computing & Applications, 31*(11), 7501–7515. doi:10.1007/s00521-018-3596-9