Int. J. Res. Ind. Eng. Vol. 8, No. 1 (2019) 1–16

International Journal of Research in Industrial Engineering
www.riejournal.com

# Multi-Objective Linear Mathematical Programming for Solving U-Shaped Robotic Assembly Line Balancing

M. Rabbani [1,*], A. H. Khezri[1], H. Farrokhi-Asl[2], S. Aghamohamadi-Bosjin[1]

[1] Department of Industrial Engineering, College of Engineering, University of Tehran, Tehran, Iran.
[2] Department of Industrial Engineering, Iran University of Science and Technology, Tehran, Iran.

**A B S T R A C T**

In recent years, robots have been an eminent solution for manufacturers to facilitate their process and focus on a variety of their products. As the importance of robot usages, our paper focuses on the robotics assembly line. In this paper, we have considered the cycle time, robot operational costs, robot purchase costs, and robot energy consumptions. In the following, we add robot failure rates to have an efficient and high-quality assembly line. The presented model is a multi-objective problem, therefore, the linear programming methods as goal programming and augmented ε-constraint method are applied to optimize the problem. In the end, we have considered a case study to examine and show the applicability of the proposed model on the real situation.

**Keywords**: Robotic mixed-model assembly line balancing (RMALB), U-shaped assembly line, Multi-objective optimization, Goal programming, Augmented ε-constraint method.

***Article history:*** *Received: 07 October 2018* *Revised: 22 January 2019* *Accepted: 26 February 2019*

## 1. Introduction

Nowadays, the competitive market leads companies to promote their manufacturing systems by more flexible and effective plan. The plan should satisfy quickly and efficiently the manufacture's demands. Due to the importance of the production plan, it was an important and controversial issue in the past decades. For the first time, Henry Ford introduced the manufacturing assembly line. Over the past years, Assembly Line Balancing (ALB) has had an eminent impact on the manufacturing systems. ALB is an ordering of a sequence stations, linked together by a transport system. Each station operates one or more tasks on the partially finished product. Based on product verity, there exist three types of assembly lines [1]:

* Corresponding author
E-mail address: mrabani@ut.ac.ir
DOI: 10.22105/riej.2018.128451.1040

- Single-Model Lines: In large quantities, only one homogeneous product is continuously produced.
- Mixed-Model Lines: On the same line in an arbitrarily inter-mixed sequence, several models of a basic product are produced.
- Multi-Model Lines: In separately batches family of products which present significant differences in processes are produced on one or more assembly lines.

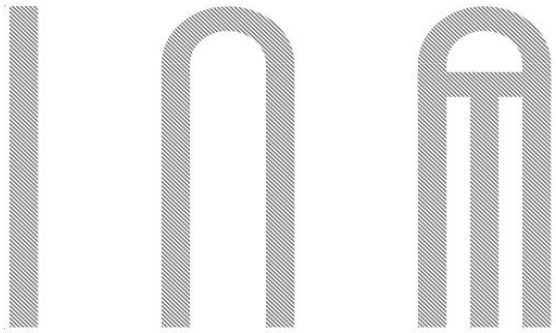

**Figure 1. Shapes of straight, U-shaped, and multi-model lines.**

In fact, ALB Problem (ALBP) lies on getting an optimal sequence of tasks to stations respect to the precedence relations among tasks and other constraints [44]. As it consumes up to 50% of total production time and accounts for more than 20% of total manufacturing cost are for assembly process. The first ALBP mathematical formulation was introduced by [2]. ALBP classified in NP-hard class [3], it is often takes a long time to find optimal solutions of large-scale problem instances by using exact solution methods. The work [4] presented a GA for solving ALBPs and later on, the [5] developed GA for solving ALBPs. This paper also focuses on the U-shaped line balancing. There are different types of assembly lines, based on the production system layout; ALBP divided into two main types: Straight and U-shaped assembly lines. In a straight line, single stations are ordered along a line, on the other hand in U-shaped line the stations are ordered within a narrow U and some of workers are allowed to perform tasks on both sides of the line (entrance of the line, exit of the line). One of the significant advantages of U-shaped lines is about providing more flexibility in assignment of tasks to stations. In a U-shaped line, the unassigned predecessors or successors are allowed to be assigned, while the straight line allows tasks to be assigned to a station if all of its predecessors have been assigned until this station. For the first time, the [6] introduced the U-Shaped Assembly Line Balancing Problem (UALBP) with developing a dynamic programming procedure for its solution. The [7] developed a mathematical formulation for UALBP-I with up to 45 tasks and yielded good results. The [8] introduced ULINO, a branch and bound procedure for solving the different types of UALBP. The [9] developed a genetic algorithm-based heuristic for the mixed-model U-shaped lines with stochastic task times. The [10] developed a Simple Genetic Algorithm (SGA) for solving the UALB-II. The [11] considered fuzzy task duration times for straight and U-shaped lines and developed a GA to solve it.

Because of the advantages and extensively uses of robots in assembly line, in this paper we have considered the Robotic Assembly Line Balancing (RALB). In the recent years, the high flexibility of robots, high productivity, and ability for production of high quality products, made robots the most prominent tools to develop the production plan. The RALB problem is the way of getting an optimal assignment to the robotic stations and selecting the best-fit robots to operate the tasks [12], and was first described by [13]. Three major objective functions are considered in RALB problem including the minimizing number of workstations (RALB-I), minimizing cycle time (RALB-II), minimizing energy consumption, and maximizing efficiency. The [14] formulated the problem for minimizing the number of workstations for a given cycle time subject to allocating of tasks to work stations. Later, the [15] extended that problem used an exact branch and bound algorithm. Also, the [16] developed an exact branch and bound algorithm.

To gain more profit and product verity of products, manufacture needs to have more flexibility with today's demanding market. In this case, the mixed-model assembly lines designed, and allowed manufacture to product a group of similar model items and provide more flexibility in producing according to market demand. Many articles have worked on the mixed-model ALBP [17- 22]. On the other hand, a few researches have focused on Robotic Mixed-Model Assembly Line Balancing (RMALB) problem. The [23] developed a mathematical model for two-sided RMALBP to minimize the cycle time respect to robot set-up and sequence-dependent set-up times. The [24] present a mathematical model for U-shaped line with considering minimizing cycle times, robot purchasing, robot set-up, sequence-dependent set-up costs. In this model, they considered two assumptions: Two or more robots can operate the work at the same station. In addition, tasks in line were handled by two groups: 1) the special tasks can be performed in one model, 2) the common tasks can be performed in several models. The [25] developed a new efficient heuristic algorithm based on beam search in order to minimize the sum of cycle times over all models.

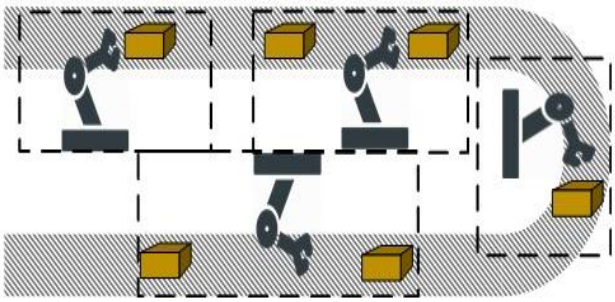

**Figure 2. U-shaped robotics assembly line form.**

## 2. Problem Description

The problem is a multi-objective type II Robotic Mixed-Model Assembly Line Balancing (RMALB-II). Based on the previous researches by [24], we have developed a new model consists of robotic operational costs and energy consumptions. In this model, we are into finding an optimal or near optimal configuration of task, workstations, and robot by considering the goals including minimization of the cycle time, robot's operational energy consumption, robots operational, and purchasing costs. Because of flexibility and adaptability of the U-shaped line, we consider it as our production line. This system allows forward and backward assignment to be performed, so the robot can move less between workstations and logically the cycle time, energy consumption, and other cost could be decreased [24, 33, 34]. To product M types of product, the U-line of assembly has J workstations with a robot in each and it has I tasks of invisible assembly task [28]. Before introducing the model, based on [28], [27], and [35] some basic assumptions considered in this paper are provided as follows:

- Power consumption of each robot is assumed and energy consumption is computed with the power consumption of each robot.
- Assembly tasks cannot be subdivided. The precedence relations among the activities are distinctive and constant. Precedence graph represented this precedence.
- The processing time of an assembly task relies on the assigned robot type that the duration of an activity by a robot is deterministic.
- Setup times between tasks are deterministic, depend on the assigned robot, and are independent of the assigned workstation.
- The line is balanced for multiple products.
- Products are models in U-shaped line.
- The purchase cost of each type robots is considered.
- The time of setup for task and robot setup times are considered.
- There are $r$ types of robot available ($r \geq 1$) that there is no restriction on the number of robots available.
- The travel times of operators are ignored.
- Material handling, loading and unloading times are insignificant, so they are contained in processing times.
- Multiple robots are allocated to each workstation.

## 2.1 Mathematical Modeling

| Indices | |
|---|---|
| $i$ | Number of assembly tasks; $i = 1, 2, \dots, I$. |
| $j$ | Number of workstations; $j = 1, 2, \dots, J$. |
| $m$ | Number of product model; $m = 1, 2, \dots, M$. |
| $r$ | Number of robot types; $r = 1, 2, \dots, R$. |

| Parameters | |
|---|---|
| $prt(i)$ | Set of immediate predecessors of task $i$. |
| $PC_r$ | Cost of a robot type $r$,to be purchased. |
| $SeC_{ri}$ | Setup cost of a robot of type $r$, for processing task $i$. |
| $SdC_i$ | Sequence dependent setup cost for processing task $i$. |
| $SeT_{ir}$ | Setup time of a robot of type $r$, for processing task $i$. |
| $SdT_i$ | Sequence dependent setup time for processing task $i$. |
| $PT_{imr}$ | Processing time of task $i$ for model $m$ by robot $r$. |
| $OEC_r$ | Operation energy consumption of the robot $r$ per time unit. |
| $SEC_r$ | Standby energy consumption of the robot $r$ per time unit. |
| $LR_r$ | Maximum length of a robot of type $r$. |
| $WR_r$ | Maximum width of a robot of type $r$. |
| $LW_j$ | Minimum length of a workstation . |
| $WW_j$ | Minimum width of a workstation $j$. |
| $CT_{max}$ | Maximum station time among all stations. |
| $EC_{max}$ | Maximum station energy among all stations. |

| Decision Variables | |
|---|---|
| $CT$ | Total cycle time. |
| $ROC$ | Robots operational costs. |
| $RPC$ | Robots purchasing costs. |
| $TEC$ | Total energy consumption. |
| $X_{ijmr}$ | 1, if the special task $i$ is assigned to the workstation $j$ and the robot type $r$ is allocated to the workstation $j$ for model $m$ of product, 0, otherwise. |
| $X_{ijr}$ | 1, if the common task $i$ is assigned to workstation $j$ and robot $r$ is assigned, 0, otherwise. |
| $Y_{jr}$ | 1, if the robot type $r$ is allocated to the workstation $j$, 0, otherwise. |
| $EC_j$ | Energy consumption of workstation $j$. |
| $CT_m$ | Cycle time for product model $m$. |

Considered objective functions are given as below:

$$Min\ CT = \sum_{m=1}^{M} CT_m \tag{1}$$

$$Min\ ROC = \left(\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{m=1}^{M}\sum_{r=1}^{R} SeC_{ri} + \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{m=1}^{M}\sum_{r=1}^{R} SdC_i\right) * (X_{ijmr} + X_{ijr}) \tag{2}$$

$$Min\ RPC = \sum_{i=1}^{I}\sum_{j=1}^{J} PC_r * Y_{jr} \tag{3}$$

$$Min\ TEC = \sum_{j=1}^{J} EC_j \tag{4}$$

$$EC_j = \sum_{r=1}^{R}\sum_{i=1}^{I}\sum_{m=1}^{M} OEC_r * PT_{imr} * X_{ijmr} + (\sum_{r=1}^{R} SEC_r * Y_{jr}) * (\sum_{m=1}^{M} CT_m - \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{r=1}^{R} PT_{imr} * X_{ijmr}) \tag{5}$$

**Table 1. Comparison table of the literature on the robotic assembly line balancing problem.**

| Article | No. of models | | Line types | | | Objectives | | | | | | | Solution Procedure |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Single | Mixed | Straight | U-shaped | Two-Sided | Min. Energy Consumption | Min. Cycle Time | Min. no. of Stations | Min. the Sequence Dependent Set Up Time | Min. no. of Robot Cell | Min. the Setup Robot | Max. Line Efficiency | |
| [13] | ✓ | | ✓ | | | | | ✓ | | | | | A branch and bound |
| [16] | ✓ | | ✓ | | | | | ✓ | | | | | Cutting plane algorithm |
| [26] | ✓ | | ✓ | | | | ✓ | | | | | | GA |
| [27] | ✓ | | ✓ | | | | ✓ | | | | | | Hybrid genetic algorithm |
| [28] | ✓ | | ✓ | | | | ✓ | | | ✓ | ✓ | | Evaluation algorithm |
| [22] | | ✓ | | | ✓ | | ✓ | | | | | | SA |
| [29] | ✓ | | ✓ | | | | | | | | | ✓ | ACO, PSO, GA |
| [30] | ✓ | | | | | ✓ | ✓ | | | | | | PSO |
| [31] | ✓ | | | ✓ | | | ✓ | | | | | | Cuckoo search, PSO |
| [32] | ✓ | | | ✓ | | | ✓ | | | | | | PSO |
| [24] | | ✓ | | ✓ | | | ✓ | | ✓ | ✓ | ✓ | | NSGA II and MOPSO |
| [14] | ✓ | | ✓ | | | | ✓ | | | | ✓ | | DE |
| [12] | ✓ | | ✓ | ✓ | | ✓ | ✓ | | | | | ✓ | DE |
| This study | | ✓ | | ✓ | | ✓ | ✓ | | ✓ | ✓ | ✓ | ✓ | Goal programming and augmented ε-contraint method |

Eq. (1) is to minimize the cycle time; Eq. (2) is to minimize the robot setup and sequence dependent setup cost of the task; Eq. (3) is to minimize robot purchasing cost; Eq. (4) is to minimize energy consumption; Eq. (5) calculates the energy consumption of each workstation consists of each stations operation energy consumption and standby energy consumption, and the considered constraints are as blow:

$$\sum_{i=1}^{I}\sum_{r=1}^{R}(PT_{imr}+SeT_{ir}+SdT_{i})*(X_{ijmr}+X_{ijr})\leq CT_{m} \qquad ,\forall j\in J,m\in M \qquad (6)$$

$$\sum_{g=1}^{J}\sum_{r=1}^{R} j*X_{hgmr}-\sum_{j=1}^{J}\sum_{r=1}^{R} j*X_{ijmr}\leq 0 \qquad ,\forall h\in prt(i),m\in M \qquad (7)$$

$$\sum_{r=1}^{R} Y_{jr}\geq 2 \qquad ,\forall j\in J \qquad (8)$$

$$\sum_{j=1}^{J}\sum_{r=1}^{R} X_{ijmr}\geq 2 \qquad ,\forall i\in I,m\in M \qquad (9)$$

$$\sum_{r=1}^{R} LR_{r}*Y_{jr}\leq LW_{j} \qquad ,\forall j\in J \qquad (10)$$

$$\sum_{r=1}^{R} WR_{r}*Y_{jr}\leq WW_{j} \qquad ,\forall j\in J \qquad (11)$$

$$\sum_{j=1}^{J}\sum_{r=1}^{R} Y_{jr}\geq 1 \qquad (12)$$

$$X_{ijmr}\in\{0,1\} \qquad ,i=1,2,\dots,I;j=1,2,\dots,J\ m=1,2,\dots,M;r=1,2,\dots,R \qquad (13)$$

$$X_{ijr}\in\{0,1\} \qquad ,i=1,2,\dots,I;j=1,2,\dots,J\ r=1,2,\dots,R \qquad (14)$$

$$Y_{jr}\in\{0,1\} \qquad ,j=1,2,\dots,J;r=1,2,\dots,R \qquad (15)$$

Eq. (6) calculates the cycle time; Eq. (7) is the precedence constraint that ensures all precedence relations among tasks between workstations; Eq. (8) shows equal or more than two robots that can be assigned to workstations. Eq. (9) indicates equal or more than two tasks that can be assigned to workstations; Eq. (10) ensures no length overlapping of robot and at each workstations, and Eq. (11) ensures no width overlapping of robot and at each workstation. Eq. (12) shows that the total number of robots used, could be more than the number of workstations. Eqs. (13-15) indicate the binary variables.

### 2.2 Failure Rate Constraint for Each Robot

In real time, by increasing in robots' processing time there would be a decreasing going in robot performances. Hence, we have considered a constraint, which limits the accepted failure ratio in each selected robot. The [36] has introduced the model as follows:

$$FR(t)_r = 1 - e^{-\lambda_r t_r}$$

$$S.t.$$

$$t_r = \sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{m=1}^{M}(PT_{imr}) * \left(X_{ijmr} + X_{ijr}\right) * Y_{jr} \qquad ,\forall r = 1, 2, \dots, R \qquad (16)$$

Where $\lambda_r$ indicates the initial failure in robot $r$; $t_r$ represents the processing time of each robot if purchase robot $r$ and assigne job $i$ in station $j$ from product $m$ to it.

## 3. Multi-Objective Mathematical Programming

In Multi-Objective Mathematical Programming (MMP), more than one objective exist to optimize and there is no single optimal solution to optimize all objectives. In these cases, finding the "most preferred" solution that would cover most of the objectives and optimize them is the solution. A multiple objective can be given in the following form [37]:

$$\begin{aligned} &Min\ F(x) = [f_1(x), f_2(x), \dots, f_m(x)] \\ &S.t. \\ &x \in X \subset IR^n \end{aligned} \qquad (17)$$

In this article, two different methods are used to find the most preferred solution: Goal Programming method and Augmented $\varepsilon$-Constraint method.

### 3.1 The Goal Programming Method

The Goal Programming (GP) is an important technique to solve Multi-Objective Decision-Making (MODM) problems in finding a set of most preferred solutions. It was first presented by [37] and later on developed by other researchers [38-41]. The main purpose of GP is to minimize the deviations between the optimal solution of each objective and their aspiration levels. It can be expressed as given model:

$$\begin{aligned} &Min \sum_{j=1}^{m} w_j * (d_j^+ + d_j^-) \\ &S.t. \quad f_j(x) - d_j^+ + d_j^- = g_j, \quad j = 1, \dots, m \\ &d_j^+ * d_j^- = 0, \quad d_j^+, d_j^- \geq 0, \quad j = 1, \dots, m \\ &x \in X. \end{aligned} \qquad (18)$$

Where $g_j$ indicates the goal for objective function $f_j$; $d_j^+$ and $d_j^-$ represent the deviation variables under and over achievement of the $j$th goal. Based on Section 0, this article model in programming model is as given:

$$
\begin{aligned}
& Min \sum_{j=1}^{5} w_j * (d_j^+ + d_j^-) \\
S.t. & \\
& CT - d_1^+ + d_1^- = CT_{min} \\
& ROC - d_2^+ + d_2^- = ROC_{min} \\
& RPC - d_3^+ + d_3^- = RPC_{min} \\
& TEC - d_2^+ + d_2^- = TEC_{min} \\
& d_j^+ * d_j^- = 0, \quad d_j^+, d_j^- \geq 0, \quad j = 1, \dots, 4 \\
& x \in X.
\end{aligned}
\tag{19}
$$

### 3.2 The Augmented $\varepsilon$-Constraint Method

The augmented $\varepsilon$-constraint method is a developed version of $\varepsilon$-constraint method and has it's own advantages. In this method, for all $p-1$ objective functions except the main one, payoff table is calculated leading to better results. To calculate the payoff table, the maximum and minimum solutions for each objective function would be calculated. Next, the range of each function is given and due to the given range, the augmented $\varepsilon$-constraint program is calculated as follows:

$$
\begin{aligned}
& Min \ f_1(x) - \delta * (\tfrac{S_2}{r_2} + \tfrac{S_3}{r_3} + \cdots + \tfrac{S_p}{r_p}) \\
S.t. & \\
& f_2 + S_2 = \varepsilon_2 \\
& f_3 + S_3 = \varepsilon_3 \\
& \cdots \\
& f_p + S_p = \varepsilon_p \\
& r_p = f_p^{max} - f_p^{min}, \quad j = 2, \dots, p \\
& \varepsilon_p = f_p^{max} - \frac{r_p}{p} * j, \quad j = 2, \dots, p\,.
\end{aligned}
\tag{20}
$$

Where $\delta$ is a small number and usually between $10^{-3}$ and $10^{-6}$; $\varepsilon_p$ is the decision makers accepted tolerance in $p$th objective function; and $j$ is total objective function interval grids points. As the presented model, therefore, the article problem in this model is as follows:

$$
\begin{aligned}
& \text{Min } \ CT - \delta * (\tfrac{S_2}{r_2} + \tfrac{S_3}{r_3} + \tfrac{S_4}{r_4}) \\
S.t. & \\
& ROC + S_2 = \varepsilon_2 \\
& RPC + S_3 = \varepsilon_3 \\
& TEC + S_4 = \varepsilon_4 \\
& r_p = f_p^{max} - f_p^{min}, \quad p = 2, 3, 4 \\
& \varepsilon_p = f_p^{max} - \frac{r_p}{p} * j, \quad p, j = 2, 3, 4\,.
\end{aligned}
\tag{21}
$$

## 4. Illustrative Example and Results Analysis

### 4.1 Test Problems

This section presents a case study of illustrated model. Consider a manufacture which has 8 kind of robots ($r = 8$) that can be purchased, producing 2 kinds of products ($m = 2$) and need to be assigned to 2 available workstations ($j = 2$) where exists 8 assembly tasks ($i = 8$) with respect to precedence processes that should be assigned to the available workstations and purchase robots. The following table represents the payoff table of objectives:

**Table 2. Payoff table of the objectives.**

| $CT\ (Sec)$ | $ROC$ (\$) | $RPC$ (\$) | $TEC\ (KW)$ |
|---|---|---|---|
| 1714 | 20000 | 1400000 | 208000 |
| 1718 | 8000 | 1400000 | 210660 |
| 1717 | 20000 | 1400000 | 207960 |
| 1722 | 30000 | 2000000 | 106260 |

In the following, we have used goal programming and augmented ε-constraint method to find the Pareto optimal solutions for each method and the results are given in following tables:

**Table 3. Goal programming method optimal Pareto solutions.**

| Solution Number | $CT$ | $ROC$ | $RPC$ | $TEC$ |
|---|---|---|---|---|
| 1 | 1715 | 8000 | 1410000 | 106260 |
| 2 | 1717 | 20000 | 2000000 | 106380 |
| 3 | 1718.9659 | 22703.1 | 2000000 | 106260 |
| 4 | 1720 | 23000 | 2000000 | 106350 |
| 5 | 1722 | 30000 | 2000000 | 106260 |

**Table 4. Augmented ε-constraint method optimal Pareto solutions.**

| Solution Number | $CT$ | $ROC$ | $RPC$ | $TEC$ |
|---|---|---|---|---|
| 1 | 1714 | 20000 | 1400000 | 208000 |
| 2 | 1714 | 20000 | 1450000 | 199575 |
| 3 | 1715 | 18000 | 1400000 | 210640 |
| 4 | 1715 | 18000 | 1500000 | 185240 |
| 5 | 1715 | 19000 | 1400000 | 209480 |
| 6 | 1716 | 14000 | 1400000 | 209640 |
| 7 | 1716 | 17000 | 1450000 | 200175 |
| 8 | 1717 | 12000 | 1450000 | 203480 |
| 9 | 1717 | 12000 | 1500000 | 187145 |
| 10 | 1717 | 13000 | 1400000 | 210400 |
| 11 | 1718 | 8000 | 1400000 | 210660 |
| 12 | 1718 | 8000 | 1500000 | 185580 |
| 13 | 1718 | 10000 | 1450000 | 202755 |
| 14 | 1718 | 11000 | 1400000 | 210500 |

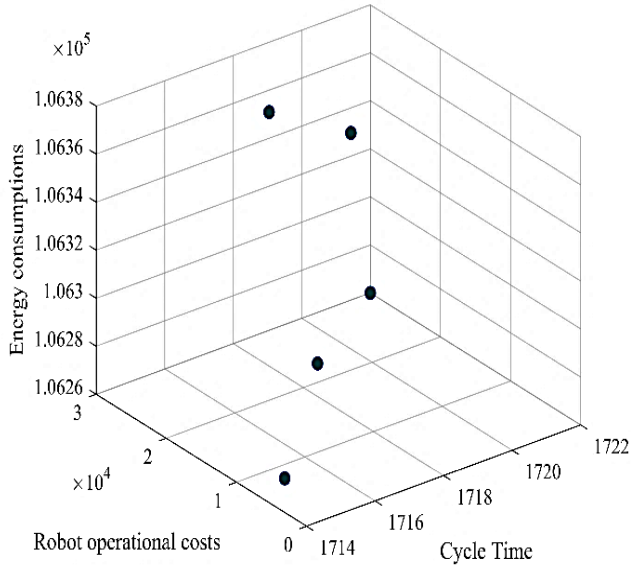


**Figure 3. Operational optimal solutions of ε-augmented constraint.**

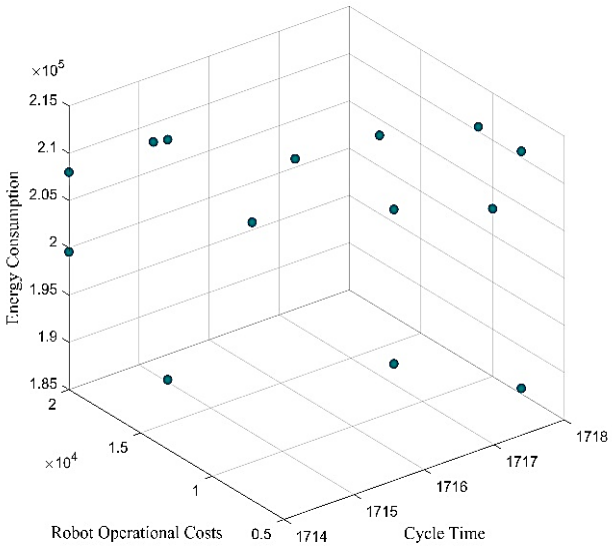


**Figure 4. Operational optimal solutions of goal programming.**

## 4.2 Comparison Metrics

In order to compare the efficiency of augmented ε-constraint method and goal programming method, the various performance metrics are reported and we have used generational number of Pareto solutions (N), distance (GD), spacing (S), and spread (Δ) in our work and explained briefly as follows:

***Generational distance***: to find a solution of Q belongs to the set of P or not, the Generational Distance (GD) evaluates an average distance of the solutions of Q from P, as follows:

$$\mathrm{GD} = \frac{(\sum_{i=1}^{|Q|} d_i^{p})^{1/p}}{|Q|}. \tag{22}$$

The parameter $d_i$ is the Euclidean distance (in the objective space) between the solution $i \in Q$ and the nearest member of $P^*$:

$$d_i = \min_{k \in |P^*|} \sqrt{\sum_{m=1}^{M} (f_m^{(i)} - f_m^{*(i)})^2}. \tag{23}$$

Where $f_m^{*(i)}$ is the $m_{th}$ objective function of the $K_{th}$ member of $P^*$. Intuitively, an algorithm hasing a small value of GD is better.

**Spacing.** The spacing metric ($S_p$) [42] is calculated with a relative distance measure between consecutive solutions in the obtained non-dominated set as follows:

$$S_p = \sqrt{\frac{1}{|Q|} \sum_{i=1}^{|Q|} (d_i - \bar{d})^2}. \tag{24}$$

Where $d_i = \min_{k \subseteq Q \wedge k \neq i} \left\{ \sum_{m=1}^{M} \left| f_m^{(i)} - f_m^{(k)} \right| \right\}$ and $\bar{d}$ is the mean value of the above distance measure $\bar{d} = \sum_{i=1}^{|Q|} {d_i}/{|Q|}$. The above metric measures the standard deviations of different $d_i$ values. When the solutions are nearly spaced, the corresponding distance measure will be small. Thus, an algorithm finding a set of non-dominated solutions has the smaller spacing (S) is better.

**Spread.** The spread metric (Δ) [43] measures the extent of spread achieved among the obtained solutions. Then, the following metric is to calculate the non-uniformity in the distribution:

$$\Delta = \frac{\sum_{m=1}^{M} d_m^e + \sum_{i=1}^{|Q|} |d_i - \bar{d}|}{\sum_{m=1}^{M} d_m^e + |Q|\bar{d}}. \quad (25)$$

Where $d_i$ is Euclidean distance between neighboring solutions that having the mean value $\bar{d}$. The parameter $d_m^e$ is the distance between the extreme solutions of $P^*$ and $Q$ corresponding to $m_{th}$ objective function. An algorithm finding a smaller value of Δ is able to find a better diverse set of non-dominated solutions.

**Table 5. Comparison of algorithms with respect to illustrative example.**

| | $N$ | $GD$ | $S_p$ | Δ |
|---|---|---|---|---|
| Augmented ε-constraint method | 14 | 29568.07245 | 0.218498741 | 0.281403455 |
| Goal programming | 5 | 240097.4139 | 0.489938335 | 0.444503484 |

## 5. Conclusion

Most of the companies that have U-shaped robotic mixed assembly line generally come across with URMALB-II in practice. Although there are many studies about assembly line balancing problems, the papers on URMALB-II are very few. Our model tries to determine the optimal or near optimal configurations with respect to considered objectives as minimizing the cycle time, robot purchasing costs, robot operational costs, and energy consumption. In this paper, we used two different multi-objective algorithms to solve the presented model. The first algorithm was weighted goal programming and the second algorithm was augmented ε-constraint method. To solve the problem, we coded the methods in GAMS. In continue, the performance of algorithms had compared with each other. As shown in Table 5, the augmented ε-constraint produced more number of Pareto points than GP. In Generational Distance (GD), GP had a better solution but in other parameters (spacing and spread), better solutions come from augmented ε-constraint. There are several interesting points for future work, developing meta-heuristic algorithms to solve the problem and considering the real life situations such as zoning constrain and proposing a dynamic model base on the online data.

## References

[1] Groover, M. P. (1980). *Automation, production systems, and computer-aided manufacturing* (Vol. 1). Englewood Cliffs, NJ: Prentice-Hall.

[2] Salveson, M. E. (1955). The assembly line balancing problem. *The journal of industrial engineering*, 6(3), 18-25.

[3] Gutjahr, A. L., & Nemhauser, G. L. (1964). An algorithm for the line balancing problem. *Management science*, *11*(2), 308-315.

[4] Falkenauer, E., & Delchambre, A. (1992, May). A genetic algorithm for bin packing and line balancing. *Proceedings of the 1992 IEEE international conference on robotics and automation* (pp. 1186-1192). Nice, France, France: IEEE.
[5] Ajenblit, D. A., & Wainwright, R. L. (1998, May). Applying genetic algorithms to the U-shaped assembly line balancing problem. *Proceedings of the 1998 IEEE international conference on evolutionary computation. IEEE world congress on computational intelligence (Cat. No. 98TH8360)* (pp. 96-101). Anchorage, AK, USA, USA: IEEE.
[6] Miltenburg, G. J., & Wijngaard, J. (1994). The U-line line balancing problem. *Management science*, *40*(10), 1378-1388.
[7] Urban, T. L. (1998). Note. Optimal balancing of U-shaped assembly lines. *Management science*, *44*(5), 738-741.
[8] Scholl, A., & Klein, R. (1999). ULINO: Optimally balancing U-shaped JIT assembly lines. *International journal of production research*, *37*(4), 721-736.
[9] Özcan, U., Kellegöz, T., & Toklu, B. (2011). A genetic algorithm for the stochastic mixed-model U-line balancing and sequencing problem. *International journal of production research*, *49*(6), 1605-1626.
[10] Jonnalagedda, V., & Dabade, B. (2014). Application of simple genetic algorithm to U-shaped assembly line balancing problem of type II. *IFAC proceedings volumes*, *47*(3), 6168-6173.
[11] Alavidoost, M. H., Tarimoradi, M., & Zarandi, M. F. (2015). Fuzzy adaptive genetic algorithm for multi-objective assembly line balancing problems. *Applied soft computing*, *34*, 655-677.
[12] Nilakantan, J. M., Ponnambalam, S. G., & Nielsen, P. (2018). Energy-Efficient straight robotic assembly line using metaheuristic algorithms. *Soft computing: theories and applications* (pp. 803-814). Singapore: Springer.
[13] Rubinovitz, J., Bukchin, J., & Lenz, E. (1993). RALB–A heuristic algorithm for design and balancing of robotic assembly lines. *CIRP annals*, *42*(1), 497-500.
[14] Nilakantan, J. M., Nielsen, I., Ponnambalam, S. G., & Venkataramanaiah, S. (2017). Differential evolution algorithm for solving RALB problem using cost-and time-based models. *The international journal of advanced manufacturing technology*, *89*(1-4), 311-332.
[15] Rubinovitz, J., & Levitin, G. (1995). Genetic algorithm for assembly line balancing. *International journal of production economics*, *41*(1-3), 343-354.
[16] Kim, H., & Park, S. (1995). A strong cutting plane algorithm for the robotic assembly line balancing problem. *International journal of production research*, *33*(8), 2311-2323.
[17] Delice, Y., Aydoğan, E. K., Özcan, U., & İlkay, M. S. (2017). A modified particle swarm optimization algorithm to mixed-model two-sided assembly line balancing. *Journal of intelligent manufacturing*, *28*(1), 23-36.
[18] Faccio, M., Gamberi, M., & Bortolini, M. (2016). Hierarchical approach for paced mixed-model assembly line balancing and sequencing with jolly operators. *International journal of production research*, *54*(3), 761-777.
[19] Faccio, M., Gamberi, M., & Bortolini, M. (2016). Hierarchical approach for paced mixed-model assembly line balancing and sequencing with jolly operators. *International journal of production research*, *54*(3), 761-777.
[20] Kara, Y., Ozcan, U., & Peker, A. (2007). An approach for balancing and sequencing mixed-model JIT U-lines. *The international journal of advanced manufacturing technology*, *32*(11-12), 1218-1231.
[21] Kucukkoc, I., & Zhang, D. Z. (2014). Simultaneous balancing and sequencing of mixed-model parallel two-sided assembly lines. *International journal of production research*, *52*(12), 3665-3687.
[22] Simaria, A. S., & Vilarinho, P. M. (2009). 2-ANTBAL: An ant colony optimization algorithm for balancing two-sided assembly lines. *Computers & industrial engineering*, *56*(2), 489-506.
[23] Aghajani, M., Ghodsi, R., & Javadi, B. (2014). Balancing of robotic mixed-model two-sided assembly line with robot setup times. *The international journal of advanced manufacturing technology*, *74*(5-8), 1005-1016.

[24] Rabbani, M., Mousavi, Z., & Farrokhi-Asl, H. (2016). Multi-objective metaheuristics for solving a type II robotic mixed-model assembly line balancing problem. *Journal of industrial and production engineering*, *33*(7), 472-484.
[25] Çil, Z. A., Mete, S., & Ağpak, K. (2017). Analysis of the type II robotic mixed-model assembly line balancing problem. *Engineering optimization*, *49*(6), 990-1009.
[26] Levitin, G., Rubinovitz, J., & Shnits, B. (2006). A genetic algorithm for robotic assembly line balancing. *European journal of operational research*, *168*(3), 811-825.
[27] Gao, J., Sun, L., Wang, L., & Gen, M. (2009). An efficient approach for type II robotic assembly line balancing problems. *Computers & industrial engineering*, *56*(3), 1065-1080.
[28] Yoosefelahi, A., Aminnayeri, M., Mosadegh, H., & Ardakani, H. D. (2012). Type II robotic assembly line balancing problem: an evolution strategies algorithm for a multi-objective model. *Journal of manufacturing systems*, *31*(2), 139-151.
[29] Daoud, S., Chehade, H., Yalaoui, F., & Amodeo, L. (2014). Solving a robotic assembly line balancing problem using efficient hybrid methods. *Journal of heuristics*, *20*(3), 235-259.
[30] Nilakantan, J. M., Huang, G. Q., & Ponnambalam, S. G. (2015). An investigation on minimizing cycle time and total energy consumption in robotic assembly line systems. *Journal of cleaner production*, *90*, 311-325.
[31] Nilakantan, J. M., Ponnambalam, S. G., & Huang, G. Q. (2015). Minimizing energy consumption in a U-shaped robotic assembly line. *2015 international conference on advanced mechatronic systems (ICAMechS)* (pp. 119-124). IEEE.
[32] Nilakantan, M. J., Ponnambalam, S. G., & Jawahar, N. (2016). Design of energy efficient RAL system using evolutionary algorithms. *Engineering computations*, *33*(2), 580-602.
[33] Manavizadeh, N., Hosseini, N. S., Rabbani, M., & Jolai, F. (2013). A Simulated Annealing algorithm for a mixed model assembly U-line balancing type-I problem considering human efficiency and Just-In-Time approach. *Computers & industrial engineering*, *64*(2), 669-685.
[34] Mukund Nilakantan, J., & Ponnambalam, S. G. (2016). Robotic U-shaped assembly line balancing using particle swarm optimization. *Engineering optimization*, *48*(2), 231-252.
[35] Li, Z., Tang, Q., & Zhang, L. (2016). Minimizing energy consumption and cycle time in two-sided robotic assembly line systems using restarted simulated annealing algorithm. *Journal of cleaner production*, *135*, 508-522.
[36] Chan, J. K., & Shaw, L. (1993). Modeling repairable systems with failure rates that depend on age and maintenance. *IEEE transactions on reliability*, *42*(4), 566-571.
[37] Charnes, A., & Cooper, W. W. (1957). Management models and industrial applications of linear programming. *Management science*, *4*(1), 38-91.
[38] Ignizio, J. P. (1985). *Introduction to linear goal programming*. Sage Publications.
[39] Lee, S. M. (1972). *Goal programming for decision analysis* (pp. 252-260). Philadelphia: Auerbach Publishers.
[40] Li, H. L. (1996). An efficient method for solving linear goal programming problems. *Journal of optimization theory and applications*, *90*(2), 465-469.
[41] Pal, B. B., Moitra, B. N., & Maulik, U. (2003). A goal programming procedure for fuzzy multiobjective linear fractional programming problem. *Fuzzy sets and systems*, *139*(2), 395-405.
[42] Schott, J. R. (1995). *Fault tolerant design using single and multicriteria genetic algorithm optimization* (Master's Thesis, Boston, MA: Department of Aeronautics and Astronautics, Massachusetts Institute of Technology).
[43] Kalyanmoy, D. (2001). *Multi objective optimization using evolutionary algorithms* (pp. 124-124). John Wiley and Sons.
[44] Kao, E. P. (1976). A preference `order dynamic program for stochastic assembly line balancing. *Management science*, *22*(10), 1097-1104.