International Journal of Industrial Engineering & Production Research June 2020 Vol. 31, No. 2: 189-202
DOI: 10.22068/ijiepr.31.2.189

RESEARCH PAPER

A Tabu Search Algorithm for the Cost-Oriented Multi-Manned Assembly Line Balancing Problem

Abdolreza Roshani[*1], **Davide Giglio**[2]

Received 15 April 2020; Revised 15 May 2020; Accepted 1 June 2020; Published online 30 June 2020
© Iran University of Science and Technology 2020

ABSTRACT
Plants manufacturing large-sized high-volume products, such as automobiles and trucks, usually encounter Multi-manned Assembly Line Balancing Problems (MALBPs). In this paper, a cost-oriented version of MALBPs, namely CMALBP, is taken into account. These types of problems may arise in the final assembly lines of products in which the manufacturing process is pretty labor-intensive. Since CMALBP is NP-Hard, a heuristic approach based on a Tabu search algorithm is developed to solve the problem. The proposed algorithm uses two neighborhood generation mechanisms, namely swap and mutation, which effectively collaborate with each other to provide new feasible solutions. Moreover, two separate tabu lists (corresponding with the two mentioned generation mechanisms) are used to check whether or not moving to a new generated neighbor solution is forbidden. To examine the efficiency of the proposed algorithm, some experimental instances were collected from the literature and solved. The obtained results show the effectiveness of the proposed tabu search approach.

KEYWORDS: *Assembly line balancing; Multi-manned workstations; Tabu search; Cost-oriented Optimization.*

1. Introduction

Assembly Line Balancing Problem (ALBP) is the problem of assigning assembly tasks to a series of workstations, arranged along a conveyor belt or a similar transportation facility, with the aim of optimizing one or more objectives when considering some restrictions that are imposed on the line. In this type of problem, each task takes a specific amount of time units to be accomplished; thus, in order to meet the desired production level, the sum of processing times of all tasks assigned to each workstation must be less than or equal to the predetermined cycle time (which is the reciprocal of the production rate). Such a restriction is called cycle time constraint. Moreover, there are precedence relations among tasks, represented by a precedence graph. In a precedence graph, each node represents a task and each arc (*i; h*) represents a precedence relation between task *i* and task *h*. In other words, (*i; h*) restricts the execution of job *h* to the execution of job *i*); this constraint is called precedence constraint. The objective function that is usually expected to balance a new assembly line is to minimize the number of workstations (line length) for a given cycle time. ALBP with this objective function is called ALBP type I. Whenever an assembly line is to be rebalanced with the aim of achieving a desired production rate, the objective function that must be considered is to minimize the cycle time for a given number of workstations (which actually corresponds to the maximization of the production rate). ALBP with this objective function is called ALBP type II. The minimization of both the number of workstations and the cycle time simultaneously characterizes the problem called ALBP type E [1].

Salveson [2] was the first researcher who presented a mathematical formulation for ALBP. His formulation has some basic assumptions: mass-production of one homogeneous product, paced-line with fixed cycle time, and serial line layout with single-manned workstations. Since

* **Corresponding author:** ***Abdolreza Roshani***
a.roshani@kut.ac.ir

1. *Department of Industrial Engineering, Faculty of Engineering Management, Kermanshah University of Technology, Kermanshah, Iran.*
2. *Department of Mechanical, Energy, Management, and Transportation Engineering (DIME), University of Genova, Genova, Italy.*

[Downloaded from www.iust.ac.ir on 2024-12-21]

then, a number of researchers have developed solution approaches to ALBP. In recent years, a majority of researchers have attempted to model more realistic and generalized problems of the assembly line balancing. The literature now contains additional characteristics such as the mixed-model production, paralleling, and multi-manned assembly lines (MAL) among many others. Detailed reviews of such studies were given by Becker and Scholl [1] and, more recently, by Battaia and Dolgui [3] and Sivasankaran and Shahabudeen [4].

Assembly line balancing problems can be classified as simple Assembly Line Balancing Problems (SALBPs) and Multi-manned Assembly Line Balancing Problems (MALBPs). Whereas only one worker is assigned to each workstation in a Simple Assembly Line (SAL), several workers can be assigned simultaneously to each workstation of an MAL. In this paper, multi-manned assembly lines were investigated. MALs are usually designed to produce high-volume large-sized standardized products such as automobiles, trucks, and buses. The reason why MALs are usually found in these kinds of production systems is because the designer of these assembly systems is allowed to assign more than one worker to each workstation due to the size of such products. The maximum number of workers that can be assigned to each multi-manned workstation is predetermined in accordance with the product size, tools availability, workstation design, etc. [5]. For instance, in automobile manufacturing, at most two workers can simultaneously work on two sides of the product. However, in assembly lines of bigger products such as trucks or buses, more than two workers simultaneously work on the product due to the size of the product and multiplicity of tasks that must be simultaneously done. According to [5], when using multi-manned workstations in such classes of assembly production systems, attempts are made to reduce the line length while the total number of workers on the line remains optimal. An MAL with a short line length provides, in practice, several advantages over a SAL as follows: shorter throughput time, lower cost of tools and fixtures, less material handling, etc. [6].

As far as this study is concerned, Dimitriadis [5] was the first researcher who dealt with the class of assembly lines at the multi-manned workstations; he also proposed a two-level heuristic-based approach to solve MALBP aiming at, first, minimizing the number of workers and, second, the number of workstations for a given cycle time. Cevikcan, Durmusoglu, and Unal [7] devised a mathematical programming model to create the assembly physical multi-manned workstations in mixed-model assembly lines; however, their proposed mathematical model was too complex to use and failed to find the desired solution to the problem. To this end, they developed a scheduling-based heuristic algorithm to solve the problem. Becker and Scholl [8] considered a special case of MALBP with Variable Workplace Parallel Assembly Line Balancing Problem (VWALBP). They assumed that the work-piece was divided into mounting positions each of which could be used by only a single worker in each workstation. In VWALBP, tasks that require at least one joint mounting position (at the same workstation) must be assigned to the same worker, i.e., after assigning a task to a worker, only the tasks which require the same or neighboring mounting positions with the assigned tasks can be assigned to that worker. The two authors developed an exact solution procedure (called VWSolver) based on the application of the branch-and-bound principle. Fattahi et al. [6] presented for the first time a mixed-integer mathematical programming model for MALBP that minimized the number of workers and workstations as its primary and secondary objectives, respectively. Since MALBP is NP-Hard, they were able to find (in a reasonable amount of time) an optimal solution by using the proposed MILP model only in the case of small-sized problems. In order to find the solution to medium- and large-sized problems, they also developed for MALBP a heuristic algorithm based on the ant colony optimization approach. Kellegoz and Toklu [9] discussed an assembly line balancing problem characterized by the presence of multi-manned workstations and, following the problem definition, developed a branch-and-bound algorithm called Jumper, to optimally solve it. Chang and Chang [10] studied the mixed-model multi-manned assembly lines and proposed a mathematical programming formulation of the problem, aiming to minimize the number of multi-manned workstations. Roshani et al. [11] addressed the multi-objective MALBP and proposed a simulated-annealing search algorithm to solve the problem while considering the line efficiency, the line length, and the smoothness index as the performance criteria. Kellegoz and Toklu [12] presented a mathematical formulation of MALBP and attempted to minimize the total number of workers on the line; they also developed a new constructive heuristic algorithm based on priority

[Downloaded from www.iust.ac.ir on 2024-12-21]

rules and a genetic algorithm-based solution procedure to improve the solutions found by the constructive heuristic. Roshani and Ghazi Nezami [13] presented the mixed-model multi-manned assembly line models and solution methods. Roshani and Giglio [14] investigated the MALBP and strived to minimize the cycle time of the line as the primary objective, for a given number of workstations. Besides the MILP model, two meta-heuristics were also developed based on SA algorithm: the indirect and direct SA (ISA and DSA, respectively). More recently, Sahin and Kellegz [15] proposed a mathematical formulation and a particle swarm optimization algorithm hybridized by a special constructive heuristic for an MAL balancing problem with a fixed cycle time that aimed to minimize both the number of workstations and the cost of the needed renewable resources.

ALBPs were categorized into two classes in accordance with the type of objective function: time-oriented ALBP and cost-oriented ALBP. Moreover, there are three variants of time-oriented ALBP, namely ALBP-I, ALBP-II, and ALBP-E, all of which aim to minimize the total idle time for the whole capacity provided by the sum of the workstations of the line; in this respect, this is called time-oriented assembly line balancing [16]. Further, the objective of cost-oriented ALBP is to minimize the total costs per production unit for a given cycle time. This objective optimizes the number of workstations by considering not only the cost of installing workstations, but also by taking into account the wage that must be payed to each worker on the line. There are some studies conducted on the cost-oriented SALBPs in the literature [16-21]. However, according to our best knowledge, only three studies appeared in the literature for Cost-oriented MALBP (CMALBP). Kazemi and Sedighi [22] presented a mathematical model and a genetic algorithm for the single model CMALBP. Their model minimized the total cost per production unit by taking into account the cost of the transportation facility per each multi-manned workstation and the wage rate of the workers; their cost function was the objective function used for balancing the simple assembly lines, too. In SALs, only one worker performs assembly tasks in each workstation; therefore, the total investment cost to be faced in installing the line is directly related to the line length, i.e., to the number of workstations equal to the number of workers. However, in MALBPs, the installing cost per each multi-manned workstation increases with an increase in the number of workers assigned to the workstation. To this end, in order to define a suitable cost function for CMALBP, it is necessary to consider the cost of tools and machinery (at each workstation) per each worker separately. Kazemi and Sedighi [23] extended their previous work to the mixed-model multi-manned assembly lines and attempted to minimize the total cost per production unit when considering the cost of tools and machinery per each worker separately. They developed a mathematical formulation of the problem and proposed a Genetic Algorithm (GA) and a particle swarm optimization algorithm to solve the problem. Roshani and Giglio [24] developed a mixed-integer programming formulation for CMALBP.

In this paper, the single model CMALBP with the aim of minimizing the total cost per production unit is investigated. A solution approach based on a tabu search algorithm is proposed to solve the problem. According to the authors' best knowledge, this is the first study that adapts a heuristic approach based on Tabu Search to solve CMALBP. The rest of this paper is organized as follows. Section 2 provides a brief description of CMALBP. Section 3 describes the proposed tabu search algorithm. Section 4 elaborates the computational studies. Section 5 concludes remarks the study.

2. Cost-Oriented Multi-Manned Assembly Line Balancing Problem

Firstly, in Subsection 2.1, the problem is defined. Secondly, a numerical example is presented in Subsection 2.2.

2.1. Problem definition

Assembly lines with multi-manned workstations are usually found in industries with large-sized and high-volume products such as automotive industry. A typical example of multi-manned assembly line configuration is shown in Figure 1. The main difference between such kinds of assembly line and SAL is the number of workers that can be assigned to each workstation. Only one worker is assigned to each workstation of a SAL, whereas several workers may be assigned simultaneously to each workstation of an MAL. The assignment of workers to a workstation of an MAL is not unregulated as the number of workers in a workstation is restricted by the maximum feasible "worker concentration" which is predetermined by the system designer in accordance with the product size, tools availability, workstation design, and so on.

[Downloaded from www.iust.ac.ir on 2024-12-21]

The present research investigates the balancing problem of the cost-oriented MALs, which is a generalization of the time-oriented MALBP-I, whose objective is to minimize the number of workstations for a given cycle time. The objective in the cost-oriented ALBPs is to minimize the total cost per product unit [19]. In general, the installation cost of each assembly line system can be divided into two groups [21]: labor costs and invested capital (like machinery and transportation facility as well as production equipment). According to [21], the workers payment (wage rate) at a workstation is specified in accordance with the job value. If a task is taken individually, it is likely to relate its job value directly to a wage rate per time unit. In other words, the payment of a worker at a workstation can be considered as the same as the wage rate of the task assigned to him/her. However, the tasks assigned to a worker may differ in the degree of difficulty, job values, and, consequently, the corresponding wage rates. In such situations, the worker's wage rate is considered as a function of the maximum job value of the tasks assigned to the workstation to which the worker has been allocated. Final assembly is, nevertheless, in need of invested capital like tools, machinery, and transportation facilities. According to [16], in a SAL, the costs associated with such resources can always be directly connected to the length of the SAL, i.e., to the number of workstations if the quantity of tools and machinery needed at workstations is assumed fixed and independent of the assignment of tasks to the workstations. Further, it is assumed that universal machinery is identical for all similar assembly tasks [21]. However, in MALs, the invested capital must be analyzed more carefully. In an MAL configuration, the costs of the transportation facility can be reliant on the length of the line (the same as in SAL), which is defined, in the case of MAL, by the number of multi-manned workstations based on the assumption that universal machinery is identical for all workstations. However, due to the different number of workers, the multi-manned workstations may need different tools and equipment to perform the assigned tasks (and, then, different costs). For instance, at a multi-manned workstation, it is likely to utilize the service of only one worker to perform the operations, while at another one, it may be necessary to assign more than one worker to the workstation; therefore, the required tools and equipment at the first workstation to which only one worker can be assigned may be smaller than the number of the required tools and equipment at the second workstation. That is why the invested capital at a multi-manned workstation should be computed not only through the universal machinery (like transportation equipment) costs, which can be considered identical for each multi-manned workstation, but also through the costs associated to the total number of tools which are required for all the workers at the workstation.

The following assumptions are given for different classes of problems considered in this study:

- the system is configured for the mass-production of one homogeneous product;
- task times are deterministic and known;
- the wage rate of each task is deterministic and known;
- the costs of the transportation facility per each workstation are fixed;
- the total costs of the machinery per each worker are fixed;
- the precedence graph is given;
- each task must be performed by a single worker;
- travel times of workers are ignored;
- parallel tasks and parallel workstations are not allowed;
- multi-manned workstation is allowed;
- the maximum number of workers that can be assigned to each workstation is given;
- workstations are aligned in a serial manner;
- the cycle time is given and fixed;
- the optimal total number of workstations and workers on the line is determined by the model;
- no further assignment restrictions are exerted besides the cycle time and precedence constraint;
- transportation facility is identical for each multi-manned workstation;
- tools and equipment are identical and independent for all workers regardless of their assigned tasks.

[Downloaded from www.iust.ac.ir on 2024-12-21]

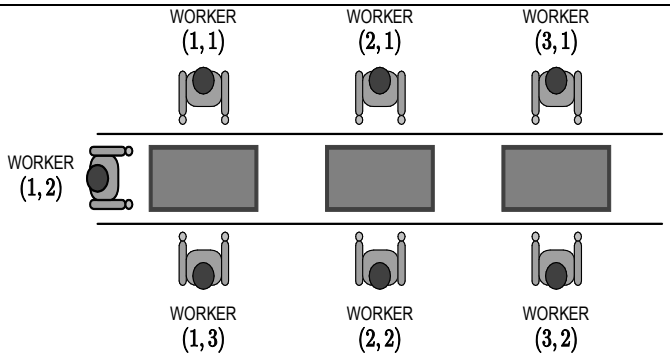

Fig. 1. Configuration of a multi-manned assembly line

2.2. Numerical example

To give a better insight into the problem, an example with respect to the CMALBP is given. Consider the precedence graph in Figure 2 presented by Bowman [25]. In this graph, each task corresponds to a node, and the links between nodes represent the precedence relation among the tasks. To each task is associated a pair of values, namely $(t_i;w_i)$, where t_i is the processing time of the i^{th} task (expressed in terms of time unit, TU) and w_i is the wage rate of the i^{th} task (expressed in terms of money units per time unit, MU/TU). Given the assumption that the cycle time is 17 TU in the example, it is possible to assign up to two workers to each workstation. Moreover, it is assumed that the cost of the transportation facility per each workstation and the cost of the machinery per each worker are 50 and 10 money units (MU), respectively. The optimal solution to this problem is obtained in the cases of both time-oriented and cost-oriented objective functions. In this connection, it should be noted that the time-oriented objective function minimizes the number of workers as the first objective and number of workstations as the second one.

Table 1 reports the optimal solutions to the two aforementioned cases.

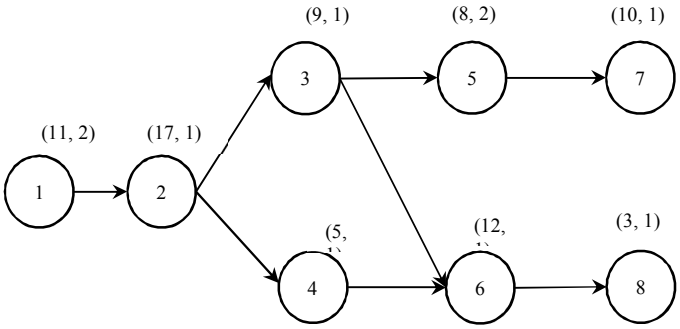

Fig. 2. Precedence diagram of an example of a multi-manned assembly line.

Tab. 1. The comparison of the optimal solution of time- and cost-oriented

		Time-oriented			Cost-oriented		
Workstation	Worker	Tasks	Wage rate	Total wage	Tasks	Wage rate	Total wage
1	1	1	2	34	1	2	34
	2	-	-	-	-	-	-
2	1	2	1	17	2	1	17
	2	-	-	-	-	-	-
3	1	3, 5	2	34	4	1	17
	2	-	-	-	3, 5	2	34
4	1	4, 6	1	17	7	1	17
	2	-	-	-	6, 8	2	34
5	1	7, 8	2	34	-	-	-
	2	-	-	-	-	-	-

[Downloaded from www.iust.ac.ir on 2024-12-21]

The summary of the results reported in Table 2 shows that the optimal number of both the workstations and workers is different in the two optimal solutions to the time-oriented and cost-oriented versions of the multi-manned assembly line balancing problem. In fact, the numbers associated with the multi-manned workstations in the first experiment (relative to the time-oriented MALBP) and the second experiment (relative to CMALBP) are 5 and 4, respectively. Whereas the number of workers is equal to 5 in the first experiment, it is equal to 6 in the second one. This difficulty increases the complexity of CMALBP with respect to the time-oriented MALBP.

Tab. 2. The comparison of the optimal solution of time- and cost-oriented (summary)

	Time-oriented	Cost-oriented
Total Wage	136	153
Workstations	5	4
Workers	5	6
Total cost	436	413

3. Proposed Solution Approach

In this paper, a Tabu Search (TS) algorithm was proposed to solve CMALBP. In fact, Gutiahr et al. [26] showed that the simple assembly line balancing problem fell into the class of NP-Hard optimization problems. In other words, the unbearable computational times prevent the determination of an optimal solution to problems of significant sizes; however, if SALBP is NP-Hard, CMALBP addressed in this paper is NP-Hard, too. Therefore, a solution approach based on a tabu search algorithm [27], which is among the most popular techniques for solving such a class of problems, is proposed. The solution approach is summarized in Figure 3.

Step 1 Initialization:
- **Step 1.1** Generate an initial solution (Y_0). Calculate $f(Y_0)$.
- **Step 1.2** Encode the generated initial solution by using the encoding scheme, thus obtain the initial solution string Π_0.
- **Step 1.3** Set the best and current solution string $\Pi_b = \Pi_c = \Pi_0$, the best and current solution $Y_b = Y_c = Y_0$, the best and the current objective function $f(Y_b) = f(Y_c) = f(Y_0)$.

Step 2 While stopping criterion is not met do:
- **Step 2.1** Set τ to 1.
- **Step 2.2** While $\tau \leq$ Neighbor-Size (NS) do:
  - **Step 2.2.1** To generate a new neighbor solution string (Π_n), apply neighbor generation mechanism on the current solution string (Π_c).
  - **Step 2.2.2** Decode Π_n, by using the proposed decoding method, in order to generate a neighbor solution Y_n.
  - **Step 2.2.3** Calculate $f(Y_n)$.
  - **Step 2.2.4** Store the neighbor solution Y_n and the corresponding value $f(Y_n)$.
  - **Step 2.2.5** Set $\tau = \tau + 1$.
- **Step 2.3** Find the neighbor solution with the best objective function among those stored at Step 2.2.4; let it be Y_n^b. If such a neighbor solution is generated by tabu move, then go to Step 2.4. Otherwise (that is, if the neighbor solution is not generated by a tabu move):
  - Set $Y_c = Y_n^b$ and $\Pi_c = \Pi_n^b$, and update the tabu list;
  - If $f(Y_n^b) - f(Y_c) \leq 0$, then set $\Pi_b = \Pi_{nb}$ and $Y_b = Y_n^b$ (thus $f(Y_b) = f(Y_n^b)$), and update the aspiration criterion.
  - Go to step 2.
- **Step 2.4** Check the aspiration criterion, that is, if $f(Y_n^b) - f(Y_b) < 0$, then set $Y_b = Y_c = Y_n^b$ and $\Pi_b = \Pi_c = \Pi_n^b$. Update the tabu list and aspiration criteria.

Step 3 Report the best solution (Y_b).

Fig. 3. The proposed tabu search algorithm

[Downloaded from www.iust.ac.ir on 2024-12-21]

The algorithm starts with a feasible initial solution (Y_0) that is generated by a heuristic approach reported in Subsection 3.1. Such a solution is initially considered as the current solution (Y_c) and the best solution (Y_b); the value of the objective function for these solutions is also calculated. The algorithm contains two main loops, namely the inner and outer loops. In the inner loop (which consists of steps from 2.2.1 to 2.2.5), the set V(Y_c) containing all neighborhood solutions Yn of the current solution Yc is determined (note that, in case V(Y_c) is too big to be fully explored, then the sub-neighborhood set V'(Yc) is determined). After exploring the neighbor solutions, the current solution, in the outer loop of the algorithm (which consists of steps from 2.1 to 2.4), is updated by the best neighborhood solution (Y_n^b) even if it is not better than the current one. This moving mechanism may cause the algorithm to be trapped in a cycle while exploring V(Y_n^b) during the next step; this phenomenon may occur because Y_n is considered the best solution regarding the V(Y_n^b), in which case we should come back to Yc and then oscillate indefinitely between Y_c and Y_n^b. To avoid such a situation (and, more generally, such cycling situations), the algorithm makes use of a tabu list TL which contains a certain number of last moves used to generate neighbor solutions. An aspiration criterion is introduced in tabu search to determine when tabu restriction can be overridden, thus removing a tabu classification otherwise applied to a move. A solution is above the current aspiration level if it is better than any solution met before.

3.1. Initial solution

Tabu Search (TS) is a local search meta-heuristic method that starts exploring the solution space from an initial solution. In this class of algorithms, the quality of the generated initial solution affects the quality of the final solution significantly. As a matter of fact, if TS starts its search from an initial solution of poor quality, it may be trapped in a local optimum. Thus, proposing an approach that can generate initial solution of good quality is necessary. In this paper, a station-oriented heuristic algorithm is proposed to find a suitable initial solution. The steps of the algorithm are as follows:

(1) Calculate an upper bound of acceptable idle time, namely $U_I = \frac{L_{wo} \cdot ct - \sum_{i \in I} t_i}{L_{wo}}$. Initiate the values of Initial Controlling Parameter (ICP) and Final Controlling Parameter (FPC).

(2) If ICP > FPC, then go to (3); else, go to (12).

(3) Let j be the counter for multi-manned workstations; set j = 0.

(4) If $j = n_w$(number of workstations), then go to (11); else set $j = j + 1$.

(5) Create a multi-manned workstation with $m = M_{max}$ workers. Let D_k be the load time of worker k; set $D_k = 0; \forall k \in K$.

(6) Determine the set $I_{np} \subseteq I$ including all tasks having no predecessors or having their predecessors assigned before.

(7) For each worker k, set the starting time S_i of all the tasks $i \in I_{np}$ to the maximum value between the completion times predecessors and D_k. Then, exclude the tasks i that violate their direct $S_i + t_i \leq ct$ for all workers at the workstation j.

(8) If $I_{np} = \emptyset$, then go to (4); else, build a roulette wheel based on the ranked positional weight of tasks and randomly select a task.

(9) Assign the selected task to the worker that can start it earlier (if tie occurs, then select a worker randomly). Go to (6).

(10) If $m = 1$, then accept the generated workload and go to (4); else, compute the mean idle time per worker, namely Id, in the current workstation, as: $I_d = \frac{m \cdot ct - \sum_{i \in I} t_i}{m}$. If $I_d \leq U_I$, then accept the generated workload; else, generate a random number r ($0 < r < 1$). If $r < ICP$, then accept the generated workload; else, set $m = m - 1$ and go to (6).

(11) Update the best initial solution. Set $ICP = \lambda \cdot ICP$ and go to (2).

(12) Stop.

Of note, both ICP and FCP take values between 0 and 1, and FCP is always less than ICP. Moreover, at Step (8) of the proposed approach, the ranked positional weight of a task is the sum of processing times of the task and of all its successors [28]. Moreover, the condition used at Step (10) was first introduced by Dimitriadis [5] with the aim of exploring the suitability of generated workloads for multi-manned workstations and also reducing the computational time of the algorithm.

3.2. Encoding

To design the tabu search algorithm for solving CMALBP, a suitable encoding scheme of a potential solution is required. In this paper, a modified version of the group-numbering encoding scheme that Kim et al. [29] used in their genetic algorithm approach was employed to solve a two-sided assembly line balancing problem. In such a representation, each solution is encoded through a string Π of length n_t (the

[Downloaded from www.iust.ac.ir on 2024-12-21]

number of tasks), and each element of Π is an integer number between 1 and *nw* (the number of workstations). Thus, if task *i* is assigned to workstation *j*, then the i^{th} element of the string is equal to *j*. For example, the string representations of the two task assignments illustrated in Table 1 are Π_1 = [1; 2; 3; 4; 3; 4; 5; 5] and Π_2 = [1; 2; 3; 3; 3; 4; 4; 4], respectively, for the time-oriented and the cost-oriented solutions.

3.3. Decoding

The encoding scheme proposed in the previous subsection specifies the assignment of tasks to the multi-manned workstations; however, it does not include any information about the assignment of tasks to the workers. Therefore, it is necessary to provide a decoding algorithm that assigns tasks to the workers and also minimizes the number of workers, taking into account the cycle time constraint and precedence relationships among the tasks. In this paper, the following heuristic method was proposed to decode the solutions.

(1) Let j be the counter for multi-manned workstations; set $j = 0$.

(2) If $j = n_w$, then go to (8); else set $j = j + 1$.

(3) Let m be the counter for the number of workers working at j; set $m = 1$.

(4) Determine the set $I_{en} \subseteq I$ including all tasks having a value of j in the encoding string.

(5) Determine the set $I_{np} \subseteq I_{en}$ including all tasks of the set I_{en} which have no precedence relation or have their predecessors assigned before. If $I_{np} \neq \emptyset$, then select the task i with the highest ranked positional weight and go to (6); else go to (7).

(6) Assign the selected task to the worker that can start it earlier (if tie occurs, then select a worker randomly). Go to (5).

(7) Calculate the completion time of all tasks assigned to workstation j, namely C_j^w, and the number of workers to whom at least one task has been assigned. If $C_j^w \leq ct$, then accept the current assignment and go to (2); else:

- if $m < M_{max}$, then let all tasks in Ien be unassigned, set $m = m + 1$, and go to (5);
- if $m = M_{max}$, then accept that assignment and go to (2).

(8) Stop.

Based on the assumption that there is always a feasible assignment of tasks to the workers for each workstation, the above algorithm assigns the tasks to the workers, taking into account both the precedence constraint (at Step (5)) and the cycle time constraints (at Step (7)). Besides, Step (7) is used to not only verify if the generated workload satisfies the cycle time constraint, but also provides a workload which minimizes the number of workers.

3.4. Neighborhood generation operators

In order to generate neighbor solutions of the current solution, two neighborhood generation mechanisms are proposed in this paper: swap and mutation.

3.4.1. Swap operator

The swap operator is applied to two randomly selected tasks and changes the value of their elements (i.e., of their workstations) in the encoded string of the current solution. The stages of swap operator procedure are given as follows.

(1) Randomly select the two tasks i and h that have no precedence relations. Read in the encoding string Πc the workstations j_i and j_h, respectively, to which they have been assigned. If $j_i = j_h$, then go to (1); else go to (2).

(2) Specify the first and the last workstations, namely j_{Fi} and j_{Li} for task i and j_{Fh} and j_{Lh} for task h, respectively, to which the selected tasks can be moved by determining the workstations of their immediate predecessors and successors. If $j_i < j_h$, then go to (3). If $j_i > j_h$, then go to (4).

(3) If $j_i \geq j_{Fh}, j_h \leq j_{Li}$, and the cycle time constraint is not violated by exchanging the workstations of the selected tasks, then set the workstation of i to j_h and workstation of h to j_i (that is, swap j_i and j_h); else go to (1).

(4) If $j_i \leq j_{Lh}, j_h \geq j_{Fi}$ and the cycle time constraint is not violated by exchanging the workstations of the selected tasks, then set the workstation of i to j_h and workstation of h to j_i (that is, swap ji and jh); else go to (1).

As an example, reconsider the precedence graph in figure 2. The string representation for the time-oriented solution of that problem is Πc = [1; 2; 3; 4; 3; 4; 5; 5]. By applying the swap operator and randomly selecting the tasks 6 and 7, the new string for generating a neighbor solution is Πn = [1; 2; 3; 3; 4; 5; 4; 5].

3.4.2. Mutation operator

The mutation operator selects a task randomly and changes its multi-manned workstation to a randomly selected multi-manned workstation it can be assigned to. The mutation operator procedure is as follows:

[Downloaded from www.iust.ac.ir on 2024-12-21]

(1) Randomly select a task i. Read in the encoding string Πc the workstation j_i to which it was assigned. Go to (2).

(2) Specify the first and the last workstations, namely j_{Fi} and j_{Li}, to which the selected tasks can be moved by determining the workstations of its immediate predecessors and successors. If $j_i = j_{Fi} = j_{Li}$, then go to (1). If $j_i = j_{Fi}$ and $j_i < j_{Li}$, then go to (3). If $j_i > j_{Fi}$ and $j_i = j_{Li}$, then go to (4). If $j_i > j_{Fi}$ and $j_i < j_{Li}$, then go to (5).

(3) Determine the set *Jct* of workstations j such that $j_i < j \leq j_{Li}$ and the cycle time constraint is not violated by transferring the selected task to them. If $J_{ct} = \emptyset$, then go to (1); else, randomly select a workstation $j \in J_{ct}$ and change the value of task i in the encoding string Πc to j.

(4) Determine the set *Jct* of workstations j so that $j_{Fi} \leq j < j_i$ and the cycle time constraint is not violated by transferring the selected task to them. If $J_{ct} = \emptyset$, then go to (1); else, randomly select a workstation $j \in J_{ct}$ and change the value of task i in the encoding string Πc to j.

(5) Determine the set *Jct* of workstations j so that either $j_{Fi} \leq j < j_i$ or $j_i < j \leq j_{Li}$, and the cycle time constraint is not violated by transferring the selected task to them. If $J_{ct} = \emptyset$, then go to (1); else, randomly select a workstation $j \in J_{ct}$ and change the value of task i in the encoding string Πc to j.

For example, consider again the precedence graph in Figure 2 and the string representation Πc = [1; 2; 3; 3; 3; 4; 4; 4]. By applying the mutation operator and randomly selecting Tasks 7 and Workstation 5 to transfer it, the new string for generating a neighbor solution is Πn = [1; 2; 3; 3; 3; 4; 5; 4].

3.5. Tabu list, aspiration criterion, and stopping rule

In this paper, two tabu lists are employed to check whether or not moving to a new generated neighbor solution is allowed. The first list, which is denoted by TL^s, is associated with the swap operator, and the second list, denoted by TL^m, is defined to control the neighbor solutions generated by the mutation operator. TL^s is a two-dimensional matrix of size $[n_t \cdot n_t]$, whose generic element TL^s_{ih} represents the number of iterations for which the swap of tasks i and h is forbidden. When the algorithm starts, TL^s is the null matrix. Whenever the best neighbor solution of current solution, generated by swapping tasks i and h, is accepted as the next current solution, values TL^s_{ih} and TL^s_{hi} are set to tabu sizes which, in this paper, are set as $\sqrt{NS}$. TL^m is instead a three-dimensional matrix of size $[n_t \cdot n_w \cdot n_w]$, whose generic element TL^m_{ijl} represents the number of iterations for which moving the task i from workstation j to workstation l is forbidden. Similar to the first list, when the algorithm starts, TL^m is a null matrix. If task i is moved from workstation j to workstation l for generating by the mutation operator, the best neighbor solution of current solution, TL^m_{ijl}, is set to the tabu size, i.e., $\sqrt{NS}$. The aspiration criterion is applied to both tabu lists. If the best neighbor solution of current solution generated by swap or mutation operator is better than the best solution, it is accepted as the current solution. The termination condition (stopping criterion) which is used in the proposed tabu search algorithm is a function of the number of iterations.

4. Computational Experiments

This section assesses the performance of the proposed TS algorithms on some well-known test problems in the literature regarding assembly line balancing problems. Five small-sized, four medium-sized, and three large-sized problems are selected. The considered dataset can be downloaded from the website "www.assembly-linebalancing.de". The efficiency and effectiveness of the proposed algorithm in case of small-sized problems are compared with the optimal solutions found by solving the MILP model presented in Section 3. Instead, since reaching the optimal solutions in a reasonable CPU time for medium- and large-sized problems by using MIP is not possible, a comparison between the results of the proposed TS and some reliable results in the literature needs to be made. As discussed in Section 1, there are two studies that have proposed solution approaches to CMALBP. However, in both of these studies, the wage rate of tasks was reported; moreover, Kazemi and Sedighi [23] presented two solution approaches to mixed-model CMALBP. Therefore, comparing the performance of the proposed algorithm with the results published in these studies is not made possible. The comparison is made with a modified version of the simulated annealing approach that Roshani et al. [11] proposed for the time-oriented MALBPs. In addition, their proposed algorithm minimizes the number of workers as the first objective, the number of workstations as the second one, and the smoothness index as the third objective for a given cycle time. The adopted modified version

[Downloaded from www.iust.ac.ir on 2024-12-21]

takes into account labor cost as considered in this paper. Thus, the aim of conducting computational experiments in the case of medium-and large-sized instances of the problem is mainly that of examining the suitability of the proposed TS algorithm in solving CMALBP in comparison to a heuristic proved to be acceptable for the time-oriented MALBP. For what concerns the determination of the optimal solutions for small-size problems, the MILP model presented by Roshani and Giglio [24] was coded and solved using Lingo 9.0 solver. The modified SA and TS algorithm were implemented in C++ language. Experiments have been carried out on a PC with a core (TM)i3 (M330) 2.13 GHz processor and 4.00 GB Ram. All the parameters of the algorithms were experimentally obtained. For small-size problems, the neighbor size and maximum number of iterations were set to 10 and 1000, respectively. Instead, in the case of medium-and large-sized problems, neighbor sizes were set to 20 and 30, and the maximum number of iterations was fixed to 3000 and 5000. Moreover, for all of the experiments, the probability of generating neighbor solutions by swap operator is 0.35 and for mutation operator is 0.65.

4.1. Small-sized problems

The results of experiments on the small-sized problems are presented in Table 3. The optimal number of workstations, optimal number of workers, optimal total cost per production unit found by the MILP model, and necessary CPU time are summarized in the mentioned table. In addition, the results of the proposed TS approach are shown. As seen in this table, TS is capable of achieving the optimal solutions for the small-sized problems in a very short period of time (less than one second).

Tab. 3. Optimal solutions of small-sized problems

			MIP				TS			
Problem	n_t	ct	n_w	n_{wo}	TC	cpu	n_w	n_{wo}	TC	cpu
Merten	7	6	3	6	360	1	3	6	360	0.09
		7	3	5	355	1	3	5	355	0.09
		8	3	5	346	1	3	5	346	0.06
		10	3	3	290	7	3	3	290	0.07
		15	2	2	230	8	2	2	230	0.07
		18	1	2	198	5	1	2	198	0.04
Bowman	8	17	4	6	625	3	4	6	625	0.09
		21	4	5	615	10	4	5	615	0.11
		24	4	5	636	3	4	5	636	0.09
		28	2	3	524	13	2	3	524	0.06
		31	2	3	501	1	2	3	501	0.06
Jaeschke	9	6	6	8	598	39	6	8	598	0.11
		7	6	7	594	1	6	7	594	0.09
		8	5	6	522	1	5	6	522	0.11
		10	3	5	410	27	3	5	410	0.09
		18	2	3	358	10	2	3	358	0.07
Jackson	11	7	6	8	663	369	6	8	663	0.14
		9	4	6	518	518	4	6	518	0.12
		10	4	5	510	151	4	5	510	0.11
		13	3	4	451	714	3	4	451	0.07
		14	3	4	440	663	3	4	440	0.11
		21	2	3	391	322	2	3	391	0.09
Mansoor	11	45	3	5	925	519	3	5	925	0.42
		54	3	4	932	170	3	4	932	0.43
		63	3	4	986	199	3	4	986	0.35
		72	2	3	952	34	2	3	952	0.39
		81	2	3	1051	283	2	3	1051	0.37

n_w: optimal number of workstations; n_{wo}: optimal number of workers; *TC*: optimal total cost per production unit; cpu: CPU time (seconds).

4.2. Medium-and large-sized problems

The performance of the proposed tabu search algorithm in solving medium- and large-sized instances of the problem is examined in this subsection. The proposed TS was applied to four medium-sized problems (respectively presented by Mitchell, Heskia, Sawyer, and Kilbridge) and three large-sized problems (one proposed by

[Downloaded from www.iust.ac.ir on 2024-12-21]

Tange, and two by Arcus), with different cycle times. Generally, 40 experiments were carried out whose results are shown in Table 4. As discussed earlier, the solutions generated by the proposed TS are not compared in this case with those provided by the MILP model (it does not provide an optimal solution in a reasonable amount of CPU time), but are compared with the modified simulated annealing algorithm version proposed by Roshani et al. [11] to solve the medium- and large-sized problems. Thus, in Table 4, the number of workstations, number of workers, and total costs per production unit found by SA are also reported. In this table, Imp value represents the improvement of the cost function from SA to TS; it is computed as $100 \cdot \frac{Tc_{SA}-TC_{TS}}{TC_{SA}}$.

Tab. 4. Computational results of medium- and large-sized problems

			SA				TS				
Problem	n_t	*ct*	n_w	n_{wo}	TC	cpu	n_w	n_{wo}	TC	*cpu*	*Imp*
Mitchell	21	14	7	8	1168	15.39	7	8	1140	1.29	2.39
		15	7	8	1185	11.18	7	8	1140	1.24	3.79
		21	5	5	1001	22.73	5	5	1001	1.23	0
		26	4	5	1054	14.37	4	5	1028	1.07	2.46
		35	3	3	980	9.38	3	3	980	0.95	0
Heskia	28	138	5	8	7708	13.9	5	8	7432	12.56	3.58
		205	4	5	8060	22.81	4	6	7340	13.23	8.93
		216	3	5	7514	20.29	4	5	7116	15.28	5.29
		256	3	5	7906	21.17	4	5	7388	12.32	6.55
		324	2	4	8028	19.32	3	4	8028	14.92	0
		342	2	3	8324	34.98	3	3	7890	13.68	5.21
Sawyer	30	25	8	14	2830	32.76	9	15	2725	13.39	3.71
		27	8	14	2840	29.84	10	13	2731	13.68	3.83
		30	7	12	2720	34.82	8	12	2710	12.59	0.36
		36	6	10	2732	58.95	7	10	2674	13.07	2.20
		41	5	9	2644	23.87	6	9	2612	23.37	1.21
		54	4	7	2662	25.89	6	7	2708	18.03	-1.7
Kilbridge	45	57	6	10	14306	65.75	6	10	14306	60.76	0
		79	4	8	11792	74.59	4	8	11634	65.96	1.34
		92	4	7	11732	83.96	4	7	10904	75.75	7.06
		110	3	6	10070	39.78	3	6	9960	38.82	1.09
		138	3	4	9416	76.17	3	4	9416	50.51	0
		184	2	3	7732	69.17	2	3	7916	54.67	-2.37
Tonge	70	176	12	22	51160	265.81	12	20	45760	149.2	10.55
		364	6	10	40848	159.78	7	10	40756	72.11	0.22
		410	5	9	38200	157.22	6	9	38790	186.25	-1.54
		468	5	8	38016	153.75	5	8	38016	146.29	0
		527	4	7	39120	159.59	4	7	38066	81.86	2.69
Arcus	83	5048	12	16	681760	346.41	16	17	657328	70.06	3.58
		5853	10	14	719536	326.11	14	14	657594	62.81	8.61
		6842	8	12	674622	204.6	10	12	657254	78.64	2.57
		7571	10	11	681822	303.2	10	11	651538	74.53	4.44
		8412	8	10	689312	296.4	10	10	674076	69.41	2.21
		8998	7	9	664862	217.4	7	9	673860	91.43	-1.35
		10816	5	8	703592	283.6	7	8	670328	83.02	4.72
Arcus	111	8847	14	18	1273500	720.3	18	19	1046780	79.71	15.41
		10027	12	16	1269210	643.6	14	18	1170920	105.7	7.74
		10743	14	15	1170040	714.1	12	15	1052610	132.4	10.03
		11378	9	14	1299200	592.1	11	15	1139530	111	12.28
		17067	7	9	1255760	593.1	9	10	1164360	85.3	7.27

n_w: optimal number of workstations; n_{wo}: optimal number of workers; *TC*: optimal total cost per production unit; cpu: CPU time (seconds).

In the case of small-sized problems, for all medium- and large-sized instances, the wage rate of tasks is fixed to an integer random number obtained from the discrete uniform distribution between 1 (MU/TU) and 10 (MU/TU). In this respect, the costs of installing each workstation and of tools and machinery per each worker they are set to 50 (MU) and 20 (MU) for Mitchell and

[Downloaded from www.iust.ac.ir on 2024-12-21]

Sawyer problems, respectively, 250 (MU) and 100 (MU) for Heskia problems; 1000 (MU) and 500 (MU) for Kilbridge and Tonge problems, respectively, and 15000 (MU) and 1000 (MU) for Arcus problems. With reference to the experiments conducted on the medium- and large-sized problems, it can be stated that TS outperforms the SA algorithm in accordance with the Imp factor which is positive (that is, TS produced a lower total cost than SA) in 30 out of 40 medium- and large-sized problem cases. In six experiments, the same result was obtained; the worst results were found by TS only in four problems. Moreover, the average improvement rate of 23 medium-sized problems and 17 large-sized problems was 2.387 and 5.26, respectively. Thus, it can be concluded that the proposed TS is more effective than SA in terms of medium-sized problems. Finally, the computation times of TS are between 0.95 and 186.25 seconds on a PC with a 2.13 GHz core i3 CPU and 4.00 GB Ram; instead, the CPU time of SA was shorter than 720.3 seconds for each problem. According to these results, it can be concluded that the proposed TS algorithm consumes a shorter computational amount of time than the SA method.

5. Conclusion

In this paper, the balancing problem of the cost-oriented multi-manned assembly lines, called CMALBP, was studied with the aim of minimizing the total cost per production unit. An illustrative example showed that given the same precedence graph of a multi-manned assembly line with the same cycle time, two different optimal solutions could be actually found when switching from the time-oriented objective function to the cost-oriented one, and vice versa. This difficulty increased the complexity of CMALBP with respect to MALBPs addressed in the literature. Since CMALBP is NP-Hard, it is impossible to find an optimal solution to the large-sized problem by using the mathematical programming formulation. Thus, a tabu search algorithm was presented to solve different sizes of the problem cases. The proposed algorithm uses a heuristic approach to generate initial solutions. Moreover, it contains two neighborhood generation mechanisms to generate new solutions and uses two different tabu lists to manage forbidden moves. The performance of the proposed TS was examined on some small-, medium-, and large-sized problem cases collected from the literature. The optimal solutions to the small-sized problems were found by coding the proposed mathematical formulation in Lingo 9.0 software; the comparison of the results showed that the proposed method could find optimal solutions. For medium- and large-sized problems, the performance of TS was compared with the simulated annealing proposed by Roshani et al. [11] for time-oriented MALBP, appropriately modified and used to solve the CMALBP. Both SA and TS were applied to the cases, and the comparison of the results of TS and SA reveals that TS is more effective than SA in terms of solution quality and computational time.

The model and solution approach proposed in this paper may be developed in future studies by considering more realistic constraints such as the sequence-dependent finish time of tasks. Moreover, since the performance of the TS approach is significantly related to the encoding algorithm, it is necessary to develop an optimal seeking procedure to optimally encode the new neighbor solutions in a reasonable amount of time.

Disclosure statement

No potential conflict of interest was reported by the authors.

References

[1] Becker, C. and Scholl, A., "A survey on problems and methods in generalized assembly line balancing." European Journal of Operational Research, Vol. 168, No. 3, (2006), pp. 694-715.

[2] Salveson, M.E. "The assembly line balancing problem." Journal of industrial engineering Vol. 6, No. 3, (1955), pp. 18-25.

[3] Battaia, O., and Dolgui, A. "A taxonomy of line balancing problems and their solution approaches." International Journal of Production Economics, Vol. 142, No. 2, (2013), pp. 259-277.

[4] Sivasankaran, P., and P. Shahabudeen. "Literature review of assembly line balancing problems." The International Journal of Advanced Manufacturing Technology, Vol. 73, No. 9-12, (2014), pp. 1665-1694.

[5] Dimitriadis, S.G., "Assembly line balancing and group working: A heuristic procedure for workers' groups operating on the same product and workstation." Computers &

[Downloaded from www.iust.ac.ir on 2024-12-21]

Operations Research, Vol. 33, No. 9, (2006), pp. 2757-2774.

[6] Fattahi, P., and Roshani, A. and Roshani, A., "A mathematical model and ant colony algorithm for multi-manned assembly line balancing problem." The International Journal of Advanced Manufacturing Technology, Vol. 53, No. 1-4, (2011), pp. 363-378.

[7] Cevikcan, E., Durmusoglu, M.B., and Unal, M.E., "A team-oriented design methodology for mixed model assembly systems." Computers & Industrial Engineering, Vol. 56, No. 2, (2009), pp. 576-599.

[8] Becker, C. and Scholl, A., "Balancing assembly lines with variable parallel workplaces: Problem definition and effective solution procedure." European Journal of Operational Research, Vol. 199, No. 2, (2009), pp. 359-374.

[9] Kellegoz, T. and Toklu, B., "An efficient branch and bound algorithm for assembly line balancing problems with parallel multi-manned workstations." Computers & Operations Research, Vol. 39, No. 12, (2012), pp. 3344-3360.

[10] Chang, H.-J., and Chang, T.-M. "Simultaneous Perspective-based Mixed-model Assembly Line Balancing Problem." Tamkang Journal of Science and Engineering, Vol. 13, (2010), pp. 327-336.

[11] Roshani, A., Roshani, A., Roshani, A., Salehi, M., and Esfandyari, A., "A simulated annealing algorithm for multi-manned assembly line balancing problem." Journal of Manufacturing Systems, Vol. 32, No. 1, (2013), pp. 238-247.

[12] Kellegoz, T. and Toklu, B., "A priority rule-based constructive heuristic and an improvement method for balancing assembly lines with parallel multi-manned workstations." International Journal of Production Research, Vol. 53, No. 3, (2015), pp. 736-756.

[13] Roshani, A., & Nezami, F. G., "Mixed-model multi-manned assembly line balancing problem: A mathematical model and a simulated annealing approach." Assembly Automation, Vol. 37, No. 1, (2017), pp. 34-50.

[14] Roshani, A., & Giglio, D., "Simulated annealing algorithms for the multi- manned assembly line balancing problem: Minimising cycle time." International Journal of Production Research, Vol. 55, No. 10, (2017), pp. 2731-2751.

[15] Sahin, M., Kellegz, T., "A new mixed-integer linear programming formulation and particle swarm optimization based hybrid heuristic for the problem of resource investment and balancing of the assembly line with multi-manned workstations." Computers & Industrial Engineering, Vol. 133, (2019), pp. 107 - 120.

[16] Amen, M., "An exact method for cost-oriented assembly line balancing." International Journal of Production Economics, Vol. 64, (2000), pp. 187-195.

[17] Rosenberg, O., Ziegler, H., "A comparison of heuristic algorithms for cost-oriented assembly line balancing." Zeitschrift fur Operations Research, Vol. 36, (1992), pp. 477-495.

[18] Amen, M., "Heuristic methods for cost-oriented assembly line balancing: A survey." International Journal of Production Economics, Vol. 68, (2000), pp. 1-14.

[19] Amen, M., "Heuristic methods for cost-oriented assembly line balancing: A comparison on solution quality and computing time." International Journal of Production Economics, Vol. 69, (2001), pp. 255-264.

[20] Scholl, A. and Becker, C., "A note on "An exact method for cost-oriented assembly line balancing"." International Journal of Production Economics, Vol. 97, (2005), pp. 343-352.

[21] Amen, M., "Cost-oriented assembly line balancing: Model formulations, solution difficulty, upper and lower bounds." European Journal of Operational Research, Vol. 168, (2006), pp. 747-770.

[22] Kazemi, A., and A. Sedighi., "A Cost-oriented Model for Multi-manned Assembly

[Downloaded from www.iust.ac.ir on 2024-12-21]

Line Balancing Problem." Journal of Optimization in Industrial Engineering, Vol. 13, (2013), pp. 13-25.

[23] Kazemi, A., and A. Sedighi., "A Cost-oriented Model for Balancing Mixed-model Assembly Lines with Multi-manned Workstations." International Journal of Services and Operations Management, Vol. 16, (2013), pp. 289-309.

[24] Roshani, A., Giglio, D., A mathematical programming formulation for cost-oriented multi-manned assembly line balancing problem. IFAC-PapersOnLine, Vol. 48, No. 3, (2015), pp. 2293-2298.

[25] Bowman, EH., "Assembly Line Balancing by Linear Programming." Operations research, Vol. 8, No. 3, (1960), pp. 385-389.

[26] Gutiahr, A.L., and Neumhauser, G.L. "An algorithm for balancing problem." Management science, Vol. 11, No. 2, (1964), pp. 308-315.

[27] Glover, F., "Tabu search: A tutorial." Interfaces, Vol. 20, No. 4, (1990), pp. 74-94.

[28] Helgeson WB., Birnie DP., "Assembly line balancing using the ranked positional weight technique." Journal of Industrial Engineering, Vol. 12, (1961), pp. 394-398.

[29] Kim, Y.K. Song, W.S. Kim, J.H., "A mathematical model and a genetic algorithm for two-sided assembly line balancing." Computers & Operations Research, Vol. 36, (2009), pp. 853-865.

Follow This Article at The Following Site:

Roshani A, Giglio D. A tabu search algorithm for the cost-oriented multi-manned assembly line balancing problem. IJIEPR. 2020; 31 (2) :189-202
URL: http://ijiepr.iust.ac.ir/article-1-1052-en.html

[Downloaded from www.iust.ac.ir on 2024-12-21]