Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2016, Article ID 9315024, 20 pages
http://dx.doi.org/10.1155/2016/9315024

Research Article

Balancing Lexicographic Multi-Objective Assembly Lines with Multi-Manned Stations

Talip Kellegöz

Industrial Engineering Department, Engineering Faculty, Gazi University, Maltepe, 06570 Ankara, Turkey

Correspondence should be addressed to Talip Kellegöz; tkellegoz@gazi.edu.tr

Received 21 October 2015; Accepted 7 March 2016

Academic Editor: Martin J. Geiger

In a multi-manned assembly line, tasks of the same workpiece can be executed simultaneously by different workers working in the same station. This line has significant advantages over a simple assembly line such as shorter line length, less work-in-process, smaller installation space, and less product flow time. In many realistic line balancing situations, there are usually more than one objective conflicting with each other. This paper presents a preemptive goal programming model and some heuristic methods based on variable neighborhood search approach for multi-objective assembly line balancing problems with multi-manned stations. Three different objectives are considered, minimizing the total number of multi-manned stations as the primary objective, minimizing the total number of workers as the secondary objective, and smoothing the number of workers at stations as the tertiary objective. A set of test instances taken from the literature is solved to compare the performance of all methods, and results are presented.

1. Introduction

Assembly lines are mostly installed for producing products in high volumes and usually include automatic material handling system [1], tools, workers, and more than one workpiece. Therefore, in fact, assembly lines are designed by arranging all of their components. One of most important problems in this arrangement is to group tasks into stations. In this problem, there are some constraints, such as precedence restrictions between task pairs and cycle time constraints for stations. This grouping process is carried out for optimizing some objectives by fulfilling all of restrictions. The optimization problem of this process is called assembly line balancing problem (ALBP).

ALBPs which are strongly NP-hard [2] can be classified based on different criteria, such as line layout (straight lines, U lines, etc.), the number of models produced (single model, multimodel, mixed model, etc.), duration of tasks (deterministic, stochastic, etc.), the number of workers for each station (traditional lines, two sided lines, multi-manned lines, etc.), and the number of objectives considered (single objective, multi-objective, etc.). The reader is referred to the recent paper by Sivasankaran and Shahabudeen [3] for a comprehensive survey and classification of line balancing problems. Also, other well-crafted surveys can be found in papers by Lusa [4], Becker and Scholl [5], and Erel and Sarin [6].

In practice, products produced in assembly lines have different characteristics based on size, the number of tasks, demand structure, duration of tasks, and so forth. Therefore, different products may require different line structures which are briefly mentioned above. For example, consider the assembly processes of a computer keyboard and a bus. It is obvious that the production of the computer keyboard requires less number of tasks than the one of bus. Also, based on sum of task times, the keyboard has smaller flow time than the bus. On the other hand, more than one worker can perform tasks on the same bus workpiece while only one worker is usually allowed to perform tasks on the same keyboard workpiece. Therefore, for decreasing the flow time, tool cost, work-in-process cost, space used for the assembly line, and so forth, assembly lines with multi-manned workstations may be more suitable to produce large size products like bus in practice. Consider the problem with

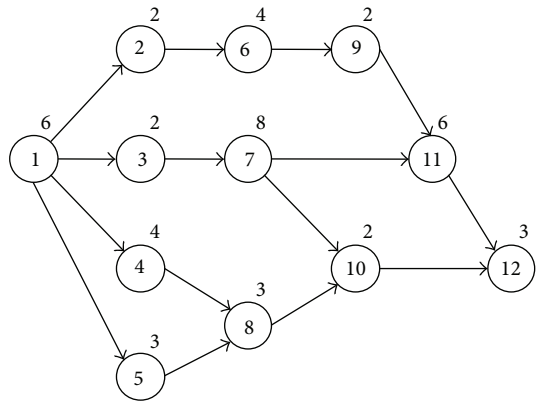


Figure 1: Precedence diagram for illustrative problem.

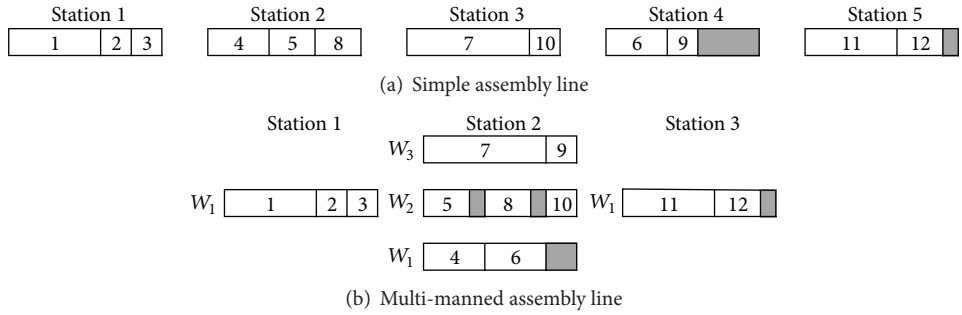


Figure 2: Optimal solutions of simple and multi-manned assembly line balancing problems.

cycle time Ct = 10 and precedence diagram given in Figure 1. By using the sum of task times $t_{\text{sum}} = 45$, the lower bound value for the total number of stations for simple assembly line [15] and also for the total number of workers for multi-manned assembly line [7, 8] can be calculated as follows:

$$\text{LB}_1 = \left\lceil \frac{t_{\text{sum}}}{\text{Ct}} \right\rceil = \left\lceil \frac{45}{10} \right\rceil = 5. \quad (1)$$

For simple assembly line balancing problem with minimizing the number of stations (SALBP1), the optimum solution is given in Figure 2(a). As is seen from this figure, at least 5 stations are required to produce the goods with desired properties. There are 5 workpieces (and workers) in the line since there is a workpiece (a worker) at each one of these stations. For multi-manned assembly line balancing problem (MALBP) with minimizing the number of stations, the optimum solution is presented in Figure 2(b). In MALBP, determining a station for each task is necessary but not sufficient. To address a MALBP solution, a worker for each task and a sequence of tasks for each worker must also be obtained. As is seen from Figure 2, the multi-manned line requires less number of stations than the simple line. Hence, in multi-manned lines, same products may be produced by using less layout space, work-in-process, flow time, and so forth.

In MALBP, each task is allowed to be performed by only one worker. Also, any worker performs at most one task at a time. To the best of authors' knowledge, the first study about MALBP was done by Dimitriadis [9]. He defined the problem and proposed a heuristic method for solving it. Then, Becker & Scholl [7] developed a mixed integer programming (MIP) model and a branch & bound (BB) method for optimally solving MALBP with mounting places. In the same study, they also described how the proposed BB is used as a heuristic for large size problem instances. For solving the mixed-model MALBP of real life tractor assembly line, Cevikcan et al. [10] proposed a heuristic method executed phase by phase in five steps. The first three steps are devoted to balance the line, and the remaining two steps are used for sequencing models and transferring workers in daily operations. For solving MALBP with minimizing both number of workers and multi-manned stations in the same order, Fattahi et al. [11] proposed a MIP model and a heuristic method based on ant colony optimization. In their comparison study, they showed their heuristic's superiority over the heuristic by Dimitriadis [9]. For solving MALBP with task times dependent on the number of workers at stations, Yazdanparast and Hajihosseini

[12] proposed a MIP model based on the model of Fattahi et al. [11]. Kellegöz and Toklu [8] considered MALBP with minimizing the number of workers and proposed a BB algorithm for optimally solving them. In terms of both CPU times and quality of feasible solutions found, comparison results showed that their BB method has better performance than the revised one of the BB method proposed by Becker and Scholl [7]. Kazemi and Sedighi [13] developed a MIP model, a genetic algorithm based heuristic and a particle swarm optimization based heuristic for cost-oriented mixed-model MALBP. They solved several test instances to illustrate their methods' performance. For solving MALBP with minimizing the total number of workers, Kellegöz and Toklu [14] recently proposed a new MIP formulation, a constructive heuristic based on priority rules and an improvement heuristic based on genetic algorithms. By using benchmark instances, they illustrated the performance of methods they proposed.

The purpose of this study is twofold: (1) to present a goal programming mathematical formulation for multi-objective MALBP and (2) to develop metaheuristic methods based on the variable neighborhood search (VNS) approach for solving medium and large size instances. To the best of authors' knowledge, this study is the first attempt to solve multi-objective MALBP by using the goal programming approach and VNS algorithm. The objectives considered for MALBP in this study are as follows: (1) minimizing the total number of multi-manned stations (line length) as the primary objective, (2) minimizing the total number of workers as the secondary objective, and (3) smoothing the number of workers at stations as the tertiary objective. The remainder of this paper is organized as follows. In Section 2, the multi-objective MALBP is defined. The goal programming formulation for multi-objective MALBP is presented in Section 3. Then, the proposed metaheuristic algorithms are presented in Section 4. The computational study and its results are reported in Section 5. At last, in Section 6, some conclusions and future research directions are discussed.

2. Multi-Objective Assembly Line Balancing Problems with Multi-Manned Stations

A multi-objective MALBP could be described as follows. There is a set I of n tasks all of them have to be performed for getting the final product. Also, some tasks' operations cannot be started before the completion of some of the remaining tasks' operations. In other words, there are precedence relationships within some task pairs. The assembly line which is built to produce this homogenous product has a set J of m stations placed on a straight line. Due to product's characteristics, each station is allowed to have more than one worker. However, the number of workers simultaneously performing different tasks at the same station cannot be greater than the value $M_{\max}$. Some reasons for this limitation may be as follows [8, 9, 11]:

(1) There are $M_{\max}$ tasks having no precedence relationship between each other and they can be performed at the same station.

(2) Due to some characteristics of the product or stations, at most $M_{\max}$ workers are able to perform tasks on the same workpiece.

All the workers used in the line are identical and equally equipped. Each task is indivisible and has to be performed for a predefined deterministic time by only one worker in the line. Each worker cannot perform more than one task at a time. However, different workers of the same station may perform different tasks simultaneously. In this case, it is also ensured that relationships between simultaneous task operations are satisfied.

The decision maker has three goals as follows (in order of priority).

Goal 1. There is no certain limitation on the line length (the number of stations on the line), but the line should not have more than $L_{\max}$ stations.

Goal 2. There is no certain limitation on the total number of workers in the line, but there should not be more than $TM_{\max}$ workers in the line.

Goal 3. There is no certain limitation, but the line should have the same number of workers in each station in the line.

It is assumed that the decision maker is not able to determine precisely the relative importance of the goals. Thus, he ranks these goals from the most important (goal 1) to least one (goal 3). As a result, the decision maker first tries to satisfy goal 1 as close as possible. Then, for all alternative optimal solutions to goal 1, he tries to find the optimal solution for goal 2, and so forth. Note that while trying to satisfy the second goal as close as possible, it is ensured that the deviation from goal 1 remains at its optimal level. Also, while coming as close as possible to meeting goal 3, it is ensured that the deviations from goals 1 and 2 remain at their optimal levels. Thus, for example, while providing goal 3, it is ensured that an additional worker that makes the second deviation worse is not used.

The reason of determining target values for the first two goals may be that the decision maker wants to make arrangement on his current line. In this case, these target values may be determined by the available space and line workforce. If there is no line active, then target values for these goals may be set to zero or their lower bound values.

Smoothness index based on station times (sum of task times assigned to a station) is calculated in the following manner [15]:

$$\mathrm{SI}_{\mathrm{st}} = \sqrt{\sum_{j=1}^{m} \left(TS_{\max} - TS_j\right)^2}, \tag{2}$$

where m is the number of opened stations in the line, TS_j is the station time of station j, and $TS_{\max}$ is the maximum station time determined as $TS_{\max} = \max_{1\le j\le m}\{TS_j\}$. Using the same approach, for multi-manned assembly lines, smoothness index based on the number of workers at stations can be calculated as follows:

$$\mathrm{SI}_{\mathrm{wr}} = \sqrt{\sum_{j=1}^{m}\left(TW_{\max} - TW_j\right)^2}, \tag{3}$$

where TW_j is the number of workers used at opened station j and $TW_{\max}$ is the maximum number of used workers at a station opened in the line, which is determined as $TW_{\max} = \max_{1\le j\le m}\{TW_j\}$. The perfect balance is indicated by smoothness index value 0. In this research, it is assumed that the last goal is satisfied by minimizing the worker based smoothness index of the line.

3. Goal Programming Formulation

The mathematical model of MALBP proposed by Fattahi et al. [11] is developed here to generate the goal programming formulation of the considered multi-objective MALBP. To the best of authors' knowledge, the term "goal programming" was introduced by Charnes and Cooper [16]. For a good description of this approach, the reader is referred to the well-known operations research textbook by Winston [17].

3.1. Notations

Parameters

Ct: Cycle time,

t_i: Duration of task i,

I: Set of tasks; $I = \{1, 2, \ldots, n\}$,

J: Set of stations; $J = \{1, 2, \ldots, n\}$, where n is a valid upper bound for the number of stations [11],

K: Set of workers at a station; $K = \{1, 2, \ldots, M_{\max}\}$,

$P(i)$: Set of immediate predecessors of task i,

$P_a(i)$: Set of all predecessors of task i,

$S(i)$: Set of immediate successors of task i,

$S_a(i)$: Set of all successors of task i,

ψ, Ω, Φ: Large positive numbers,

$L_{\max}$: Target value of the number of stations opened in the line,

$TM_{\max}$: Target value of the total number of workers used in the line.

Decision Variables

st_i: Start time of task i,

$$x_{ijk} = \begin{cases} 1, & \text{if task } i \text{ is assigned to the worker } k \text{ at station } j \\ 0 & \text{otherwise,} \end{cases}$$
$$y_{ih} = \begin{cases} 1, & \text{at the same worker, if task } i \text{ is performed before tash } h \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

w_j: Number of workers used at station j,

mw: The maximum of number of workers of stations,

$mw = \max_j\{w_j\}$,

$$\alpha_{jk} = \begin{cases} 1, & \text{if station } j \text{ is opened, and it has } (mw - k) \text{ workers} \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$
$$u_j = \begin{cases} 1, & \text{if station } j \text{ is opened} \\ 0 & \text{otherwise,} \end{cases}$$

ll: Total number of stations opened in the line (line length),

d_1^+, d_1^-: Positive and negative deviations from the target value $L_{\max}$ of the number of stations opened in the line,

d_2^+, d_2^-: Positive and negative deviations from the target value $TM_{\max}$ of the total number of workers used in the line,

d_3^+, d_3^-: Positive and negative deviations from the target value zero of the smoothness index value of the line (without square root).

3.2. Mathematical Model. The proposed mixed integer goal programming model is as follows.

3.2.1. Objective Function. Consider

$$\text{lexmin}\left\{d_1^+, d_2^+, d_3^+\right\}. \tag{6}$$

3.2.2. Model Constraints

General Constraints. Consider

$$\sum_{j\in J}\sum_{k\in K} x_{ijk} = 1, \quad \forall i \in I, \tag{7}$$

$$\text{Ct}\cdot\sum_{j\in J}\sum_{k\in K}\left((j-1)\cdot\left(x_{hjk}-x_{ijk}\right)\right)+\text{st}_h+t_h\le\text{st}_i, \quad \forall i\in I,\ h\in P(i), \tag{8}$$

$$\text{st}_i + t_i \le \text{Ct}, \quad \forall i \in I, \tag{9}$$

$$\text{st}_h+\Psi\cdot\left(1-x_{hjk}\right)+\Psi\cdot\left(1-x_{ijk}\right)+\Psi\cdot\left(1-y_{ih}\right)\ge\text{st}_i+t_i, \quad \forall i\in I,\ h\in\left\{r\mid r\in I-\left(P_a(i)\cup S_a(i)\right),\ i<r\right\},\ j\in J,\ k\in K, \tag{10}$$

$$\text{st}_i+\Psi\cdot\left(1-x_{hjk}\right)+\Psi\cdot\left(1-x_{ijk}\right)+\Psi\cdot y_{ih}\ge\text{st}_h+t_h, \quad \forall i\in I,\ h\in\left\{r\mid r\in I-\left(P_a(i)\cup S_a(i)\right),\ i<r\right\},\ j\in J,\ k\in K. \tag{11}$$

Constraints for Goal 1. Consider

$$\sum_{j\in J}\sum_{k\in K} j\cdot x_{ijk}\le ll, \quad \forall i\in I, \tag{12}$$

$$ll + d_1^- - d_1^+ = L_{\max}. \tag{13}$$

Constraints for Goal 2. Consider

$$\sum_{k\in K} k\cdot x_{ijk}\le w_j, \quad \forall i\in I,\ j\in J, \tag{14}$$

$$\sum_{j\in J} w_j + d_2^- - d_2^+ = TM_{\max}. \tag{15}$$

Constraints for Goal 3. Consider

$$mw \ge w_j, \quad \forall j\in J, \tag{16}$$

$$\Omega\cdot u_j\ge\sum_{i\in I}\sum_{k\in K} x_{ijk}, \quad \forall j\in J, \tag{17}$$

$$\Phi\cdot\left(1-u_j\right)+\sum_{k\in k}\left(k\cdot\alpha_{jk}\right)+w_j\ge mw, \quad \forall j\in J, \tag{18}$$

$$\sum_{k\in K}\alpha_{jk}\le 1, \quad \forall j\in J, \tag{19}$$

$$\sum_{j\in J}\sum_{k\in K}\left(k^2\cdot\alpha_{jk}\right)+d_3^- - d_3^+ = 0. \tag{20}$$

Sign Restrictions. Consider

$$x_{ijk}\in\{0,1\}, \quad \forall i\in I,\ j\in J,\ k\in K, \tag{21}$$

$$y_{ih}\in\{0,1\}, \quad \forall i\in I,\ h\in\left\{r\mid r\in I-\left(P_a(i)\cup S_a(i)\right),\ i<r\right\}, \tag{22}$$

$$\alpha_{jk}\in\{0,1\}, \quad \forall j\in J,\ k\in K, \tag{23}$$

$$u_j\in\{0,1\}, \quad \forall j\in J, \tag{24}$$

$$w_j\ge 0, \quad \text{and integer } \forall j\in J, \tag{25}$$

$$\text{st}_i\ge 0, \quad \forall i\in I, \tag{26}$$

$$d_1^+, d_1^-, d_2^+, d_2^-, d_3^+, d_3^-, ll, mw \ge 0. \tag{27}$$

Constraints (7), (9)–(11) are directly taken from the model which is proposed by Fattahi et al. [11]. Constraint (8) is written by using the clock time approach [15]. The proposed mixed integer goal programming model has multiple objectives given in Expression (6). The first goal is that the number of opened stations should not exceed predetermined value $L_{\max}$. As indicated earlier, the variable d_1^+ corresponds to overachievement of this target value. Therefore, the first objective aims to minimize this positive deviation. The second objective aims to minimize the positive deviation of total number of workers used in the line from predetermined target value $TM_{\max}$. At last, the third objective aims to minimize the positive deviation of smoothness index on the basis of number of workers at stations (without square root). Note that the target value of smoothness index is equal to zero. Constraint (7) ensures that each task is assigned to only one worker of the line. Constraint (8) ensures that all precedence relationships among tasks are satisfied. This constraint can also be written as follows:

$$\text{Ct}\cdot\sum_{j\in J}\sum_{k\in K}(j-1)\cdot x_{hjk}+\text{st}_h+t_h \le \text{Ct}\cdot\sum_{j\in J}\sum_{k\in K}(j-1)\cdot x_{ijk}+\text{st}_i, \quad \forall i\in I,\ \forall h\in P(i). \tag{28}$$

Constraint (9), cycle time constraint, ensures that each task is completed before the end of the cycle time. If tasks i and h which have no precedence relationship between each other are assigned to the same worker of the same station, then one of Constraints (10) and (11) becomes active (with an integer value $\Psi \ge \text{Ct}$). While the activation of Constraint (10) by using $y_{ih} = 1$ ensures $\text{st}_h \ge \text{st}_i + t_i$, the activation of Constraint (11) by using $y_{ih} = 0$ ensures $\text{st}_i \ge \text{st}_h + t_h$. Constraints (12) and (13) are used to formulate the first goal. Constraint (12) assigns proper value to variable ll. Constraint

(13) is the goal programming constraint of the number of multi-manned stations with deviational variables. Having d_1^+ with the value zero ensures that the total number of stations is $L_{\max}$ or less. The set of Constraints (14) and (15) models the second goal. Constraint (14) assigns proper value to variable w_j. Constraint (15) with deviational variables is the goal constraint of the total number of workers used in the line. While the value of d_2^+ is reduced to zero, the number of workers used in the line decreases towards the value $TM_{\max}$. The last set having Constraints (16)–(20) corresponds to the third goal. Constraint (16) ensures that the value of mw is equal to or greater than the number of workers used in each station. Constraint (17) with an integer value $\Omega \geq n$ ensures that if station j has at least one assigned task (i.e., it is opened), then $u_j = 1$. Constraints (18) and (19) are used to assign proper value to each α_{jk} variable. Constraint (18) with an integer value $\Phi \geq M_{\max}$ is active when the corresponding station is opened. For station j, note that Constraint (19) does not allow more than one α_{jk} variable to have the value 1. When Constraints (18) and (19) with the third objective are examined together, it can be seen that all α_{jk} variables of unopened station j will have the value zero. Constraint (20) is the goal programming constraint of the smoothness index on the basis of number of workers at stations. Note that the first term of this constraint computes the sum of squared deviations (from the maximum of number of workers at stations, mw) of the number of workers of stations. While the value of d_3^+ is reduced to zero, the smoothness index value decreases towards zero. This constraint is written based on the fact that minimizing the value of a positive valued function also minimizes the value of the square root of this function. Constraints (21)–(25) show the integer variables, and Constraints (26) and (27) indicate that all the remaining variables are nonnegative continuous.

3.3. Solving the Proposed Model. In the proposed model, there are three goals as follows (in order of importance):

$$\begin{aligned} &\text{Minimize } G_1 = d_1^+ \,(\text{Highest priority}) \\ &\text{Minimize } G_2 = d_2^+ \,(\text{Mid-level priority}) \\ &\text{Minimize } G_3 = d_3^+ \,(\text{Lowest priority}). \end{aligned} \tag{29}$$

This model is solved by considering one goal at a time, starting with goal 1 and terminating with goal 3. In a solution process, it is ensured that a lower priority goal never degrades any higher priority objective function value. For doing this in a solver, the current solution is added as a new constraint to problems with lower priorities. For example, consider d_1^* is the optimum objective value of the first problem. Then, the constraint $d_1^+ = d_1^*$ is added to the second problem. After solving the second problem, if the optimum objective value is d_2^*, then both constraints $d_1^+ = d_1^*$ and $d_2^+ = d_2^*$ are added to the last problem. After solving the last problem, the optimum solution of the goal programming model is found [18].

For all alternative optimal solutions to the highest-level objective, the solution which optimizes the second level is found and so on. That is, deviations from the goals are minimized in order of priority. Thus, the resulting solution is called the lexicographical minimum [19].

4. Proposed Variable Neighborhood Search Heuristics

VNS was introduced by Mladenović [20]. It combines local search with systematic changes of neighborhood for escaping from local optimum in the descent [21]. The basic characteristic of VNS is that it searches over several neighborhood structures in the solution space. Hence, it reduces the solution space into smaller subspaces.

The basic VNS procedure [22] is given in Algorithm 1. At each iteration of VNS search process, there is an incumbent solution which is a local optimum, and VNS searches in solutions which are in a far neighborhood of the current incumbent solution. From the current neighborhood, it randomly determines a solution (shaking phase), applying local search to this solution. If the resulting solution is better than the current incumbent solution, then the search moves to this new solution. If not, a farther neighborhood structure (after the farthest one, the nearest structure) is used to go on the search process.

Contrary to other metaheuristics based on local search methods, VNS does not follow a trajectory but explores increasingly distant neighborhoods of the current incumbent solution and jumps from this solution to a new one if and only if an improvement has been made [21]. VNS heavily relies upon the following observations [23]:

(i) A local minimum with respect to one neighborhood structure is not necessarily a local minimum for an another neighborhood structure.

(ii) A global minimum is a local minimum with respect to all possible neighborhood structures.

(iii) For many problems, local minima with respect to one or several neighborhoods are relatively close to each other.

For solving the multi-objective MALBP with VNS, following components of VNS have to be developed: (1) solution coding and encoding schemes, (2) the method used for generating initial solution, (3) neighborhood structures used in shaking phase, (4) local search procedure, and (5) the method used to compare solutions (objective function). Thus, in the next subsections, these components for solving multi-objective MALBP are described in detail.

4.1. Representing Solutions (Coding). Solutions of the multi-objective MALBP are represented by using permutations of tasks' priorities. In other words, for the problem with n tasks, let $(p_{i_1}, p_{i_n}, \ldots, p_{i_n})$ be a permutation of the value n. Then

Initialization
(i) Select the set of neighborhood structures N_k, $k = 1, 2, \ldots, k_{\max}$, that will be used in the search;
(ii) Find an initial solution x;
(iii) Choose a stopping condition;
Main step: Repeat the following sequence until the stopping condition is met:
(1) Set $k \leftarrow 1$;
(2) Until $k = k_{\max}$, repeat the following steps:
(a) *Shaking*: Generate a point x' at random from the kth neighborhood of x ($x' \in N_k(x)$);
(b) *Local search*: Apply some local search method with x' as initial solution; denote x'' the so obtained local optimum.
(c) *Move or not*: If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and continue the search with N_1 ($k \leftarrow 1$); otherwise, set $k \leftarrow k + 1$;

Algorithm 1: The basic VNS procedure.

Procedure - Build Solution
begin
(1) *BestSol* ← *null*;
(2) Create initial line configuration, *LConf*;
(3) *CurrSol* ← assign tasks to line with configuration *LConf*;
(4) **if** (*CurrSol* is better than *BestSol*) **then**
(4.1) *BestSol* ← *CurrSol*;
(4.2) *BaseConf* ← get line configuration of *CurrSol*;
(4.3) *LR* ← *list* of *BestSol*'s stations having more than one used worker;
end if;
(5) **if** (*LR* = {}) **then**
(5.1) **return** *BestSol*;
end if;
(6) *s* ← first element in *LR*;
(7) Delete the first element in *LR*;
(8) *LConf* ← *BaseConf*;
(9) Decrease the number of workers at station *s* of *LConf* by 1;
(10) Go to Step (3);
end;

Algorithm 2: Procedure of solution construction process.

the priority values for tasks $i_1, i_2, \ldots, i_n$ are $p_{i_1}, p_{i_n}, \ldots, p_{i_n}$, respectively. These priority values are used to determine which one of available tasks is selected for assignment at each step. This approach which is called priority based coding is one of commonly used structures for other types of assembly line balancing problems in the literature [24, 25].

4.2. Building Solutions (Encoding). For building a solution based on priority values, a restricted version of the constructive heuristic proposed by Kellegöz and Toklu [14] is used. The procedure of building solution used in this study is given in Algorithm 2.

Second step in the procedure generates an initial line configuration and sets it to the current configuration *LConf*. A line configuration is a vector of the same length as the maximum number of stations, and the element *i* in this vector determines how many workers are available at the station *i* of the line. The initial line configuration has the same number of workers at stations with the best one of solutions in the set IS. If the solution has no worker at a station, then the corresponding element in the line configuration is set to $M_{\max}$. For comparing solutions, the method described in the next subsection is used. The set IS includes $M_{\max}$ solutions, and the solution in *s*th element is created by the following way. First, a line configuration having the number of *s* workers in each one of *n* stations is created. As indicated earlier, *n* is the valid upper bound value for the number of stations. Then, the method described in the following paragraph is used to assign tasks to stations. In this manner, it is aimed to start the solution building process from a good line configuration.

Assigning tasks to a line with a known line configuration is carried out in Steps (2) and (3). First, an empty station with the number of workers which is determined from the

line configuration is opened. At each assignment step, a task having the following properties (checked in the same order) is selected: (1) it is not assigned yet, (2) all its predecessors are already assigned, (3) it can be completed before the end of cycle time, (4) it can be started earlier than other tasks having previous properties, and (5) it has greatest priority value within tasks having previous properties. The selected task is assigned to the worker who can start the processing of this task earlier. If more than one worker has the same minimum start time, tie is broken by selecting the worker having the smallest index. If there are unassigned tasks, and none of these tasks cannot be assigned to the current station, then a new empty station having the number of workers determined from the line configuration is opened.

After getting the current solution *CurrSol*, it is compared with the best solution *BestSol* found in the solution building process. The method which is used to compare solutions is presented in the next subsection. If the current solution is better than the best solution, then (1) the best solution is updated, (2) the line configuration *BaseConf* is created, and (3) the list *LR* is determined. The line configuration *BaseConf* is determined by using the newly found best solution. This line configuration has n stations and includes the same number of workers at each opened station of the best solution (active part of the configuration) and $M_{\max}$ workers at each one of other stations which is not opened in the best solution (passive part of the configuration). The list *LR* which includes best solution's stations which have more than one worker (multi-manned) is used in solution restriction process. This list includes stations based on minimum worker load-related strategy proposed by Kellegöz and Toklu [14]. According to this strategy, stations in *LR* are decreasingly ordered according to their minimum worker's load.

As is seen from the solution building procedure presented in Algorithm 2, if the list *LR* has no element, then the solution building process is ended. If not stopped, then the configuration restriction process is performed by using Steps (6)–(9). First, the first element in the list *LR* is determined (s), and then it is deleted from the list *LR*. After this process, the current line configuration *LConf* is set to the configuration *BaseConf*. Then, the procedure decreases the number of workers at station s of *LConf* by 1. By this way, it is aimed to reduce the number of workers used in the line which is the secondary objective of the considered MALBP. Note that the main function for searching the best solution based on all objectives is performed by VNS based heuristics proposed.

For illustrating the solution construction process, consider the priority permutation (5, 12, 3, 10, 9, 1, 6, 2, 7, 11, 4, 8) for the problem instance with cycle time 10, $M_{\max} = 3$, and precedence diagram given in Figure 1. All three target values are considered zero. In this case, three different line configurations (a row vector with $n = 12$ elements) are considered for determining initial line configuration: (1) each element is set to the value 3 (i.e., $M_{\max}$), (2) each element is set to the value 2 (i.e., $M_{\max} - 1$), and (3) each element is set to the value 1 (i.e., $M_{\max} - 2$). In order to illustrate the task assignment process, the assignment process for the first line configuration is summarized in Table 1. Resulting solutions for the determination process of the initial line configuration are presented in Figure 3. As is seen from this figure, resulting solutions have 3, 4, and 5 stations, respectively. Also, they have 7, 6, and 5 workers and 4, 2, and 0 worker based smoothness indexes (without square root), respectively. When the solution comparison method given in the next subsection is considered, it is concluded that the best of these three solutions is the one given in Figure 3(a). This solution has 3, 3, and 1 workers for stations 1, 2, and 3, respectively. Hence, the initial line configuration is determined as (3, 3, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3).

After creating the current solution and updating the best solution, the base line configuration *BaseConf* is determined as (3, 3, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3) and the list *LR* is formed as (Station 1, Station 2). Note that Station 3 is not added to this list since it has only one used worker. After executing Steps (6)–(9) of the procedure, the configuration *LConf* is determined as (2, 3, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3), and the list *LR* is updated as (Station 2). The assignment process for this *LConf* configuration and the resulting solution are given in Table 2 and Figure 4(a), respectively. In this case, the current solution has the same number of stations with the best solution and less number of workers than the best solution. Therefore, according to the solution comparison method given in the next subsection, this newly generated solution is better than the best solution. Thus, after updating the best solution, the base line configuration *BaseConf* and the list *LR* are set to (2, 3, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3) and (Station 1, Station 2), respectively. After executing Steps (6)–(9) of the procedure, the configuration *LConf* is determined as (1, 3, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3), and the list *LR* is updated as (Station 2). The resulting solution for this configuration is given in Figure 4(b). As is seen from this figure, the resulting solution is again better than the best one. Thus, the best solution is updated. Then, the base line configuration *BaseConf* and the list *LR* are set to (1, 3, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3) and (Station 2), respectively. After executing Steps (6)–(9) of the procedure again, the configuration *LConf* is determined as (1, 2, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3), and the list *LR* is updated as { }. The resulting solution for this configuration is given in Figure 4(c). As is seen from this figure, the resulting solution is not better than the best one. The process is stopped since there is no element in the current list *LR*.

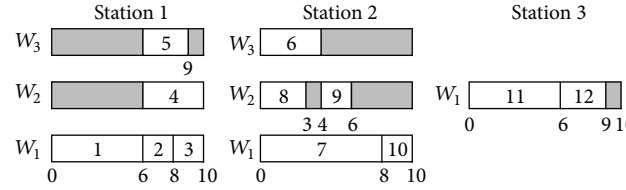
4.3. Comparing Solutions. A simple comparison method is not applicable for comparing solutions since there is more than one objective in the considered problem. Comparison method has to include all objectives with the same priority order. Thus, for comparing two different solutions (Sol_1 and Sol_2), the algorithm which has the flowchart given in Figure 5 is used. Notations s_1 and s_2 represent the number of stations opened in Sol_1 and Sol_2, respectively. Also, notations m_1 and m_2 represent the total number of workers used in Sol_1 and Sol_2, respectively, and notations ns_1 and ns_2 represent worker based smoothness index of Sol_1 and Sol_2, respectively. As is seen from Figure 5, if both solutions have the total

Table 1: Task assignment process for the line configuration (3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3).

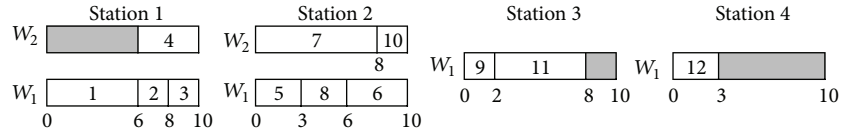
Step	Station index (man count)	Task with no predecessor (priorities)	Selected task	Start time	Man index	Notes
1	1 (3)	1 (5)	1	0	1	—
2		2 (12), 3 (3), 4 (10), 5 (9)	2	6	1	—
3		3 (3), 4 (10), 5 (9), 6 (—)	4	6	2	Task 6 cannot be completed before the end of cycle time.
4		3 (3), 5 (9), 6 (—)	5	6	3	Task 6 cannot be completed before the end of cycle time.
5		3 (3), 6 (—), 8 (—)	3	8	1	Tasks 6 and 8 cannot be completed before the end of cycle time.
6		6 (—), 7 (—), 8 (—)	—	—	—	(i) Tasks 6, 7, and 8 cannot be completed before the end of cycle time. (ii) New station is opened.
7	2 (3)	6 (1), 7 (6), 8 (2)	7	0	1	—
8		6 (1), 8 (2)	8	0	2	—
9		6 (1), 10 (—)	6	0	3	Task 10 cannot be started at time 0.
10		9 (7), 10 (—)	9	4	2	Task 10 cannot be started at time 4.
11		10 (11), 11 (—)	10	8	1	Task 11 cannot be completed before the end of cycle time.
12		11 (—)	—	—	—	(i) Task 11 cannot be completed before the end of cycle time. (ii) New station is opened.
13	3 (3)	11 (4)	11	0	1	—
14		12 (8)	12	6	1	—

Table 2: Task assignment process for the line configuration (2, 3, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3).

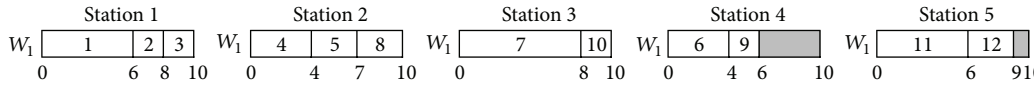
Step	Station index (man count)	Task with no predecessor (priorities)	Selected task	Start time	Man index	Notes
1	1 (2)	1 (5)	1	0	1	—
2		2 (12), 3 (3), 4 (10), 5 (9)	2	6	1	—
3		3 (3), 4 (10), 5 (9), 6 (—)	4	6	2	Task 6 cannot be completed before the end of cycle time.
4		3 (3), 5 (—), 6 (—)	3	8	1	Tasks 5 and 6 cannot be completed before the end of cycle time.
5		5 (—), 6 (—), 7 (—)	—	—	—	(i) Tasks 5, 6, and 7 cannot be completed before the end of cycle time. (ii) New station is opened.
6	2 (3)	5 (9), 6 (1),7 (6)	5	0	1	—
7		6 (1), 7 (6), 8 (—)	7	0	2	Task 8 cannot be started at time 0.
8		6 (1), 8 (—)	6	0	3	Task 8 cannot be started at time 0.
9		8 (2), 9 (—)	8	3	1	Task 9 cannot be started at time 3.
10		9 (7), 10 (—)	9	4	3	Task 10 cannot be started at time 4.
11		10 (11), 11 (—)	10	8	1	Task 11 cannot be completed before the end of cycle time.
12		11 (—)	—	—	—	(i) Task 11 cannot be completed before the end of cycle time. (ii) New station is opened.
13	3 (1)	11 (4)	11	0	1	—
14		12 (8)	12	6	1	—



(a) Solution for the line configuration (3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3)

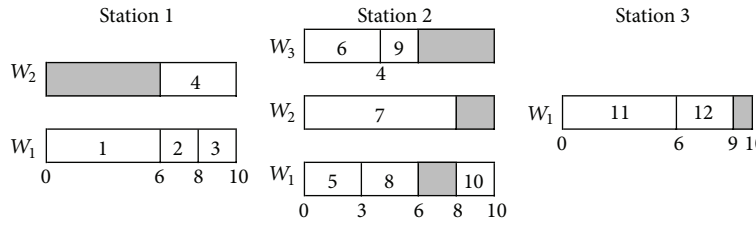


(b) Solution for the line configuration (2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2)

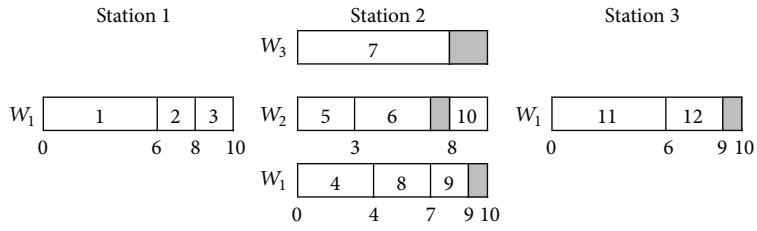


(c) Solution for the line configuration (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)

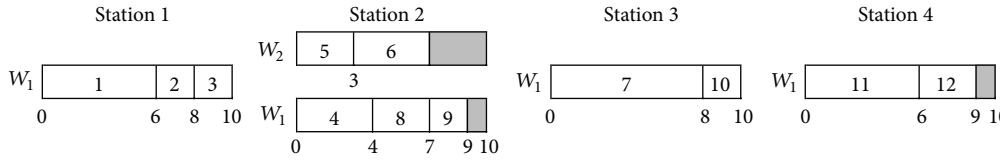
Figure 3: Resulting solutions for the determination of the initial line configuration.



(a) Solution for the line configuration (2, 3, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3)



(b) Solution for the line configuration (1, 3, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3)



(c) Solution for the line configuration (1, 2, 1, 3, 3, 3, 3, 3, 3, 3, 3, 3)

Figure 4: Illustration of the solution building process.

number of opened stations less than or equal to the target value $L_{\max}$, or both solutions have the same total number of opened stations, then it is decided that there is no difference between two solutions according to the total number of opened stations. The same approach is applied for testing solutions based on both the total number of used workers and worker based smoothness indexes. Note that the target value for smoothness index is always zero, and the check is carried out in the same order with objectives. If there is no difference between two solutions based on three criteria, then

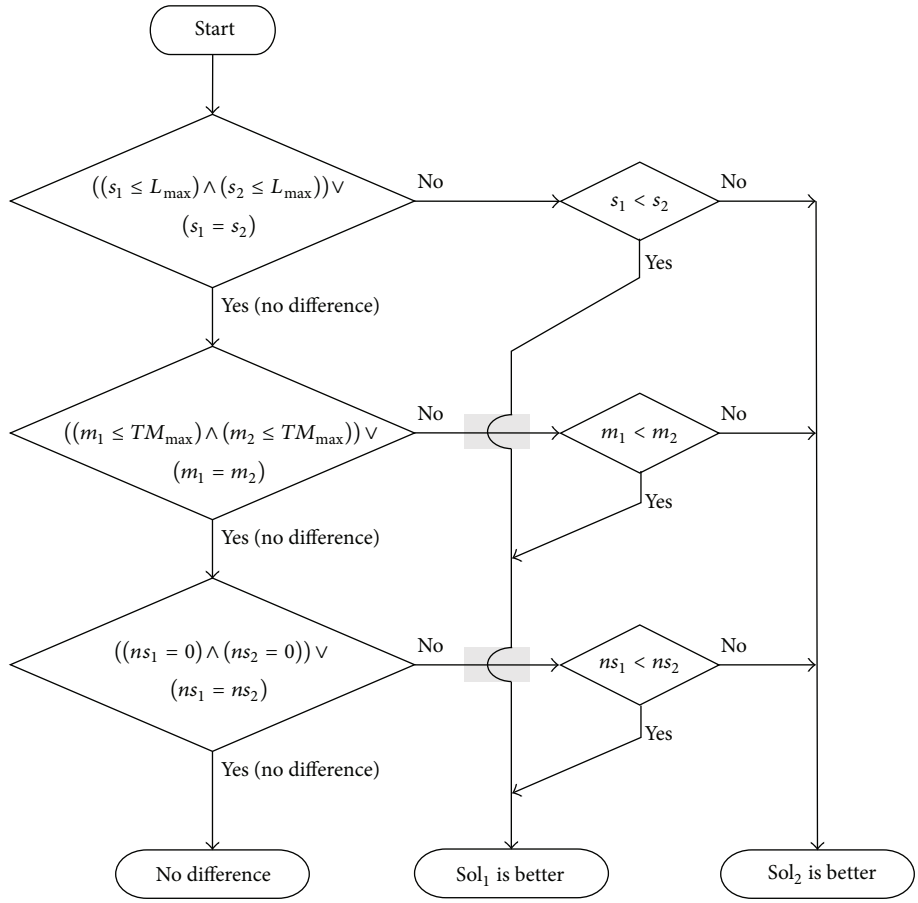


Figure 5: Flowchart of procedure comparing two solutions.

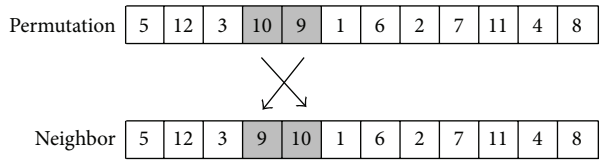


Figure 6: Illustration for producing a neighbor from API neighborhood.

it is decided that there is no difference between two solutions. If there is a difference based on one of three criteria, then it is concluded that the solution having less value based on this criterion is better.

4.4. Neighborhood Structures. For generating neighbors, the following two methods are used in this paper:

(i) adjacent pairwise interchange (API),

(ii) subset randomization (SR).

In a permutation, two adjacent digits are interchanged to generate an API neighbor. This process is illustrated in Figure 6. There are $n-1$ neighbors in API neighborhood of a permutation of the value n [26].

For generating a random neighbor by using SR mechanism, a subset of digits is first selected randomly. Then, unselected digits are copied to the same positions of the neighbor. Last, selected digits are randomly copied to empty positions of the neighbor. For SR structure, it is obvious that there is more than one neighborhood based on the number of digits selected randomly. For generating a neighbor different from its origin, at least two digits have to be selected. Let k and SR^k be the number of randomly selected digits and the neighborhood structure with parameter k, respectively. For a permutation of the value n, the interval of the parameter k is $2 \le k \le n$. The SR^2 ($k = 2$) method generates neighbors from general pairwise interchange neighborhood (PI) with probability 0.5. The remaining permutations are the same with their origins. Therefore, for the origin solution x, $SR^2 = PI \cup \{x\}$ is valid. On the other hand, the SR^n method generates random permutations of n. An illustrative example for the method SR^3 is presented in Figure 7.

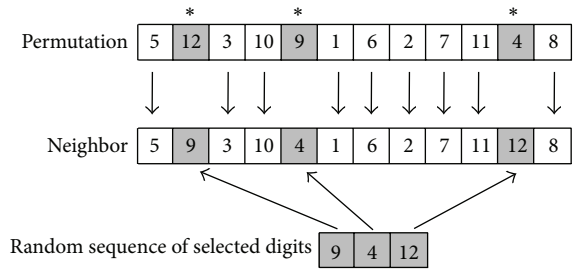


Figure 7: Illustration for producing neighbor from SR^3 neighborhood.

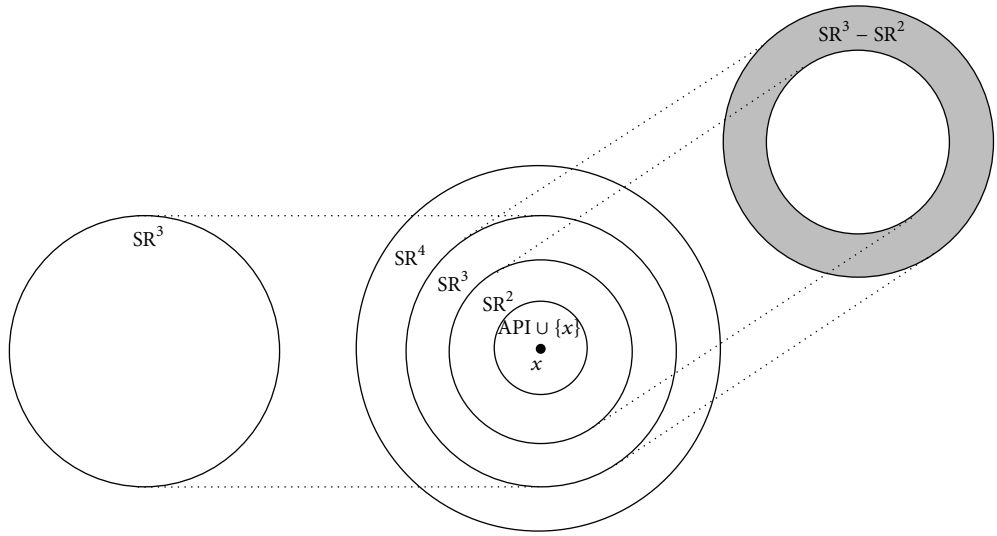


Figure 8: Relationship between neighborhoods.

If only $k-1$ digits of a neighbor which is generated from SR^k has different values from its origin, then this neighbor is also an element of SR^{k-1}. In this case, this neighbor has the same value with its origin at only one of selected k digits. Likewise, if a neighbor has the same values with its origin at two of selected k digits, then this neighbor is also an element of both SR^{k-1} and SR^{k-2}. Thus, the relationship $SR^{k-1} \subset SR^k$ is valid. This relationship is illustrated in Figure 8. If a neighbor $x' \in SR^k$ has a different value from its origin x at each one of k digits, then $x' \notin SR^{k-1}$ is valid. In this case, $x' \in (SR^k - SR^{k-1})$ is also valid. As a result, in neighborhood generation process, ensuring that each one of k positions has different value from its origin also ensures that the generated neighbor is an element of $SR^k - SR^{k-1}$.

4.5. Variations of VNS for Multi-Objective PMALBP. For solving large size problem instances, getting a good initial solution is important. For these instances, local search algorithms often require substantial amounts of running time. Hence, Hansen and Mladenović [27] introduced a new version of VNS called Reduced VNS (RVNS). For decreasing computation time, RVNS heuristic omits the local search step of the basic VNS algorithm. In this paper, RVNS heuristic is also considered.

Hansen et al. [22] declared that decomposition might be worthwhile in heuristics for very large problem instances since the performance of VNS depends in general on the local search subroutine used. Based on this approach, they proposed a new VNS based algorithm called Variable Neighborhood Decomposition Search (VNDS). VNDS decomposes the problem space by fixing all but k attributes in the local search phase for a given solution. For the considered problem in this paper, the solution attributes are priority values of tasks. Thus, priority values of tasks which are not changed in the shaking phase are determined, and the same priority values are fixed in the local search phase. Due to this fixation, the local search phase is carried out by using API structure on a restricted part of the solution (restricted API). For the SR^3 neighbor solution given in Figure 7, all but priority values of tasks 2, 5, and 11 are fixed. Thus, local search phase is applied based on only these three digits. The restricted API neighborhood for this SR^3 solution is given in Figure 9.

In brief, we developed three different heuristics based on VNS variations for solving the multi-objective MALBP.

Table 3: Properties of VNS variations.

Property	VNS	RVNS	VNDS
Initial solution	(i) Generate random. (ii) Apply local search with API structure.	(i) Generate random. (ii) Apply local search with API structure.	(i) Generate random. (ii) Apply local search with API structure.
Shaking phase	$SR^2 - (API \cup \{x\})$ $SR^3 - SR^2$ $\vdots$ $SR^{K_{max}} - SR^{K_{max}-1}$	$SR^2 - (API \cup \{x\})$ $SR^3 - SR^2$ $\vdots$ $SR^{K_{max}} - SR^{K_{max}-1}$	$SR^2 - (API \cup \{x\})$ $SR^3 - SR^2$ $\vdots$ $SR^{K_{max}} - SR^{K_{max}-1}$
Local search phase	API	—	Restricted API
Value of the parameter K_{max}	$\max\{2, \lceil 0.5 \cdot n \rceil\}$	$\max\{2, \lceil 0.7 \cdot n \rceil\}$	$\max\{2, \lceil 0.8 \cdot n \rceil\}$

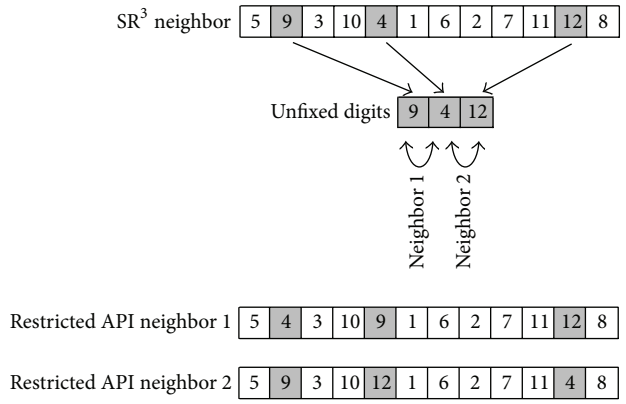


Figure 9: Generating neighbors in the local search phase of VNDS.

These heuristics are summarized in Table 3. For initiating proposed algorithms with a good incumbent solution, local search with API structure is applied to a solution which is generated randomly.

As is seen from Table 3, all methods use only one parameter, K_{max}, which is the top level of SR neighborhood structures used in the heuristic's shaking phase. For identifying the value of this parameter for each method, we carried out preliminarily experiment by using different size preliminary test instances. The K_{max} parameter value is set to each one of $\max\{2, \lceil r \cdot n \rceil\}$ values where n is the number of tasks in the problem instance and r is a control parameter getting a value from the domain $\{0.1, 0.2, \ldots, 1.0\}$. By using each one of r values, ten different K_{max} values are determined based on the size of considered instance. For each parameter value for an instance, each one of VNS, RVNS, and VNDS is run five times. After examining the results of each method, a superior level of the K_{max} parameter is determined. These selected levels are also presented in Table 3.

5. Computational Study

In order to evaluate the proposed MIP and heuristics, experiments were conducted by using benchmark instances. All heuristics were coded in C# language and compiled in the programming software Microsoft Visual C# 2010 Express Edition. MIP models of instances were solved by using the solver IBM ILOG CPLEX (version 12.5.1.0). All experiments were run on a PC with i3 predecessor, 6 GB RAM, and 64 bit Microsoft Windows 7 operating system.

Benchmark instances were generated by the following way. Test problems (task times and precedence relationships) with cycle times were given from the literature. Three small size data sets are P11, P21, and P30. They have 11, 21, and 30 tasks and are obtained from the papers by Jackson [28], Mitchell [29], and Sawyer [30], respectively. Moreover, some medium and large size data sets were also used in experiments. These data sets are P45 [31], P70 [32], P83 and P111 [33], and P148 [34]. Two different levels, 2 and 4, were considered for the parameter M_{max} (admitted maximum number of workers at a station). For each data set, three different values for the cycle time parameter were obtained from the literature. In the value determination process for target value parameters of instances, it was aimed to compare the performance of proposed methods based on the following manners:

(i) minimizing the number of stations, the number of workers, and smoothness index in the same order,

(ii) minimizing the number of workers, and smoothness index in the same order due to given target value of the number of stations,

(iii) minimizing smoothness index due to given target values of the number of stations and workers.

The method which was used to determine target values is presented in the next subsections. For an instance with n tasks, each heuristic was run for $3 \cdot n$ CPU seconds. Also, for every experimental instance, each heuristic was applied 5 times. Each execution of MIP models has a time limit of 3,600 CPU seconds.

5.1. Minimizing the Number of Stations, the Number of Workers, and Smoothness Index in the Same Order. A target value for a variable means that the decision maker desires

Table 4: Results for small and medium size instances for minimizing the number of stations, the number of workers, and smoothness index in the same order.

Problem	Ct	M_{max}	VNS result	RVNS result	VNDS result	MIP result	MIP CPU
P11	7	2	6[a]; 8[b]; 4[c]	6; 8; 4	6; 8; 4	6; 8; 4	0.16; 0.14; 0.33
		4	5; 9; 10	5; 9; 10	5; 9; 10	5; 9; 10	0.14; 0.11; 0.08
	10	2	4; 5; 3	4; 5; 3	4; 5; 3	4; 5; 3	0.14; 0.11; 0.28
		4	3; 6; 5	3; 6; 5	3; 6; 5	3; 6; 5	0.19; 0.11; 0.09
	21	2	2; 3; 1	2; 3; 1	2; 3; 1	2; 3; 1	0.11; 0.09; 0.22
		4	2; 3; 1	2; 3; 1	2; 3; 1	2; 3; 1	0.16; 0.17; 0.2
P21	14	2	7; 8; 6	7; 8; 6	7; 8; 6	7; 8; 6	0.62; 0.75; 0.37
		4	7; 8; 6	7; 8; 6	7; 8; 6	7; 8; 6	0.87; 0.52; 0.89
	21	2	4; 6; 2	4; 6; 2	4; 6; 2	4; 6; 2	0.92; 0.64; 0.58
		4	4; 6; 2	4; 6; 2	4; 6; 2	4; 6; 2	2.18; 1.94; 0.86
	35	2	3; 3; 0	3; 3; 0	3; 3; 0	3; 3; 0	0.67; 1.17; 0.2
		4	3; 3; 0	3; 3; 0	3; 3; 0	3; 3; 0	1.47; 1.94; 0.33
P30	25	2	8; 14; 2	8; 14; 2	8; 14; 2	8; 14; 2	9.1; 56.94; 1.36
		4	8; 14; 2	8; 14; 2	8; 14; 2	8; 14; 2	10.64; 122.27; 6.3
	30	2	7; 12; 2	7; 12; 2	***6.6*; 12; 1.2**	6; 12; 0	103.49; 80.65; 0.89
		4	6; 12; 10	**6; 12; *8***	6; 12; 10	6; 12; 0	16.66; 2822.2; 62.54
	41	2	***4.8*; 8.2; 1.4**	***4.8*; 8; 1.6**	5; 8; 2	4; 8; 0	1741.97; 358.52; 10.05
		4	4; 9; 3	**4; *8.8*; 3.6**	4; 9; 3	4; 8; No	26.52; 594.38; 2979.15
P45	57	2	6; 10; 2	6; 10; 2	6; 10; 2	6[int]; —; —	3600.24; —; —
		4	5; 10; 7	5; 10; 7	5; 10; 7	5; 10[int]; —	121.4; 3479.07; —
	110	2	3; 6; 0	3; 6; 0	3; 6; 0	3; 6[int]; —	315.87; 3284.37; —
		4	3; 6; 0	3; 6; 0	3; 6; 0	3; 6[int]; —	215.03; 3385.43; —
	184	2	2; 3; 1	2; 3; 1	2; 3; 1	2; 4[int]; —	167.22; 3433.02; —
		4	2; 3; 1	2; 3; 1	2; 3; 1	2; 3[int]; —	99.17; 3501.32; —
P70	176	2	12; 22; 2	12; 22; 2	**12; *21.6*; 2.4**	No; —; —	3600.27; —; —
		4	9; 22.8; 6.2	9; 23; 5.6	**9; *22.4*; 6.6**	70[int]; —; —	3601.36; —; —
	364	2	6; 10; 2	6; 10; 2	6; 10; 2	No; —; —	3600.27; —; —
		4	**4; *10.8*; 2.8**	4; 11; 1	4; 11; 1	No; —; —	3600.32; —; —
	468	2	4; 8; 0	4; 8; 0	4; 8; 0	No; —; —	3600.25; —; —
		4	3; 9; 0	**3; *8.8*; 0.2**	3; 9; 0	3; 9[int]; —	1168.64; 2432.65; —

[a]Mean number of stations.
[b]Mean number of workers.
[c]Mean smoothness index (without square root).
[int]Integer solution was found within a given CPU time limit.
No: no solution was found within a given CPU time limit.
—: phase was not performed since previous phases had consumed the given CPU time.

this variable to have a value not greater than the target value. In this case, for minimization of a variable, there is no difference between values equal to or smaller than the target value. From this point on, for minimizing the number of stations, the number of workers, and smoothness index in the same order, all target values are set to zero. Results for all heuristics and MIP models are given together in Table 4. Results of heuristics in this table are presented based on mean values. Note that results of MIP models are also equal to deviation values since all target values are equal to zero. For each instance, heuristic results are compared with each other based on the comparison method given in Figure 5. As is seen from Table 4, there is no difference between all methods for instances of P11 and P21 since all heuristics found optimal solutions. For one instance (with Ct = 41 and $M_{max} = 4$) in the problem group P30, the MIP formulation was not able to find the optimal solution. Also, the remaining instances (P45 and P70) could not be solved by using the MIP formulation. For example, for the instance P45 with Ct = 57, and $M_{max} = 2$, the MIP formulation was only able to find an integer solution for goal 1 within the given CPU time limit. Hence, the remaining phases for the remaining

Table 5: Results for large size instances for minimizing the number of stations, the number of workers, and smoothness index in the same order.

Problem	Ct	M_{max}	VNS result	RVNS result	VNDS result
P83	5048	2	10[a]; 16[b]; 4[c]	10; 16; 4	10; 16; 4
		4	**9; *17.2*; 12**	9; 17.4; 14.4	9; 17.4; 13.2
	6842	2	7.6; 12.4; 2.8	7.4; 12.6; 2.2	7.6; 12.4; 2.8
		4	7; 13; 5.8	**7; 13; *5.4***	7; 13; 7.6
	7571	2	6.2; 11; 1.4	**6; 11; 1**	**6; 11; 1**
		4	6; 11; 1	6; 11; 1	6; 11; 1
P111	5755	2	15.8; 28.4; 3.2	***15.6*; 28.2; 3**	15.8; 28.2; 3.4
		4	12.2; 29.4; 36	***12*; 29.4; 42.2**	***12*; 30; 41.6**
	8847	2	11; 18; 4	11; 18; 4	11; 18; 4
		4	10; 18.2; 27.4	**10; *18*; 21.2**	**10; *18*; 22**
	10743	2	8.4; 15; 1.8	***8*; 15; 1**	***8*; 15; 1**
		4	7; 15.6; 28.4	**7; *15.4*; 24.4**	**7; *15.4*; 25.2**
P148	434	2	7; 14; 0	7; 14; 0	7; 14; 0
		4	5; 14; 13.2	***4.4*; 14; 2.4**	4.8; 14; 1.6
	626	2	5; 10; 0	5; 10; 0	5; 10; 0
		4	3; 10; 2	3; 10; 2	3; 10; 2
	805	2	4; 8; 0	**4; *7.8*; 0.2**	4; 8; 0
		4	2.6; 8; 0.6	***2*; 8; 0**	2.2; 8; 0.2

[a]Mean number of stations.
[b]Mean number of workers.
[c]Mean smoothness index (without square root).

goals were not carried out (sign "–"). For the instance P45 with Ct = 57, and M_{max} = 4, the MIP formulation was able to find the optimum solution for goal 1 within 121.4 CPU seconds. Therefore, phase 2 was started. In phase 2, only an integer solution could be found within the given CPU time limit (121.4 + 3479.07 = 3600.47 CPU seconds). Hence, the last phase was not carried out. The overall evaluation of the results found in this group is as follows. All heuristics were able find optimal solutions for 14 instances whose optimal solutions were found by the MIP formulation. There are 8 instances for which heuristics showed different performance. For 2 of them (25%), VNS was able to find better solutions than at least one of other heuristics. RVNS and VNDS found better solutions for 4 (50%) and 3 (37.5%) instances, respectively.

Results for large size instances are presented in Table 5. There are 18 instances in this group. All heuristics found the same solution for each of 7 instances. For one of the remaining 11 instances, VNS was able find better solution than at least one of other two heuristics (9%). On the other hand, RVNS found better solutions for 10 instances (91%). Lastly, VNDS has better performance than at least one of other heuristics for 5 instances (46%). For these 5 instances, note that RVNS found the same solutions with VNDS. Hence, VNDS outperformed only VNS for this group of instances.

5.2. Minimizing the Number of Workers and Smoothness Index in the Same Order due to Given Target Value of the Number of Stations. As stated earlier, there is no difference between values which are in the desired interval determined by a target value for a variable. For example, if the target value for the number of stations is set 4, then values 2, 3, and 4 are not superior against each other. On the other hand, in this case, these values are superior against values greater than 4, and the value 5 is superior against values greater than 5 and so forth. For comparing the performance of heuristics according to the minimization of the number of workers and smoothness index in the same order, a target value of the number of stations for each instance was determined. Care has been taken to ensure that all heuristics could reach target values. Hence, for an instance, a target value of the number stations was set to the maximum of 15 values found in Section 5.1. Note that each of three heuristics was applied 5 times for an instance. Due to goals of minimizing the number of workers and smoothness index, target values for these goals were set to zero.

The results of the computational experiments for small and medium size instances are presented in Table 6. As is seen from this table, instances having more than 30 tasks could not be solved by the MIP formulation. For instances in sets P11, P21, and P45, there is no difference between heuristics according to their results. For the problem group P30, VNS and VNDS produced better solutions for one instance and two instances, respectively. A similar observation may be done for the problem group P70 since VNDS emerges as the best method for this group. Shortly, for 6 medium size

Table 6: For a given target value of the number of stations, results for small and medium size instances for minimizing the number of workers, and smoothness index in the same order.

Problem	Ct	$M_{\max}$	Target # stations	VNS result	RVNS result	VNDS result	MIP		
							d_1^+	Result	CPU
P11	7	2	6	6[a]; 8[b]; 4[c]	6; 8; 4	6; 8; 4	0	6; 8; 4	0.12; 0.16; 0.33
		4	5	5; 9; 10	5; 9; 10	5; 9; 10	0	5; 9; 10	0.16; 0.11; 0.09
	10	2	4	4; 5; 3	4; 5; 3	4; 5; 3	0	4; 5; 3	0.11; 0.16; 0.28
		4	3	3; 6; 5	3; 6; 5	3; 6; 5	0	3; 6; 5	0.09; 0.13; 0.11
	21	2	2	2; 3; 1	2; 3; 1	2; 3; 1	0	2; 3; 1	0.13; 0.13; 0.22
		4	2	2; 3; 1	2; 3; 1	2; 3; 1	0	2; 3; 1	0.08; 0.19; 0.2
P21	14	2	7	7; 8; 6	7; 8; 6	7; 8; 6	0	7; 8; 6	0.59; 0.75; 0.38
		4	7	7; 8; 6	7; 8; 6	7; 8; 6	0	7; 8; 6	2.22; 0.53; 0.92
	21	2	4	4; 6; 2	4; 6; 2	4; 6; 2	0	4; 6; 2	0.94; 0.66; 0.56
		4	4	4; 6; 2	4; 6; 2	4; 6; 2	0	4; 6; 2	2.57; 2.01; 0.84
	35	2	3	3; 3; 0	3; 3; 0	3; 3; 0	0	3; 3; 0	0.61; 1.17; 0.17
		4	3	3; 3; 0	3; 3; 0	3; 3; 0	0	3; 3; 0	2.36; 1.75; 0.28
P30	25	2	8	8; 14; 2	8; 14; 2	8; 14; 2	0	8; 14; 2	13.12; 56.16; 1.37
		4	8	8; 14; 2	8; 14; 2	8; 14; 2	0	8; 14; 2	6.04; 122.3; 6.29
	30	2	7	7; 12; 2	7; 12; 2	**6.6; 12; *1.2***	0	7; 12[int]; —	7.55; 3592.52; —
		4	6	6; 12; 10	6; 12; 10	6; 12; 10	0	6; 12; 0	16.37; 3052.52; 67.28
	41	2	5	5; 8.2; 1.8	4.8; 8.2; 1.4	**4.6; *8*; 1.2**	0	5; 8; 0	12.61; 467.86; 42.62
		4	4	**4; *8.8*; 2.4**	4; 9; 3	4; 9; 3	0	4; 8; No	10.94; 639.06; 2950.07
P45	57	2	6	6; 10; 2	6; 10; 2	6; 10; 2	0	6; 10[int]; —	253.77; 3346.47; —
		4	5	5; 10; 7	5; 10; 7	5; 10; 7	0	5; 10[int]; —	73.91; 3526.56; —
	110	2	3	3; 6; 0	3; 6; 0	3; 6; 0	0	3; 6[int]; —	54.09; 3546.15; —
		4	3	3; 6; 0	3; 6; 0	3; 6; 0	0	3; 6[int]; —	95.38; 3505.09; —
	184	2	2	2; 3; 1	2; 3; 1	2; 3; 1	0	2; 3[int]; —	39.28; 3560.96; —
		4	2	2; 3; 1	2; 3; 1	2; 3; 1	0	2; 3[int]; —	60.23; 3540.24; —
P70	176	2	12	12; 22; 2	12; 22; 2	12; 22; 2	No	No; —; —	3600.29; —; —
		4	9	**9; *22.8*; 10.8**	9; 23; 6	**9; *22.8*; 10**	0	9; 22[int]; —	497.19; 3104.08; —
	364	2	6	6; 10; 2	6; 10; 2	6; 10; 2	No	No; —; —	3600.27; —; —
		4	4	4; ***11***; 1	4; 11.2; 0.8	**4; *11*; 1**	0	4; 12[int]; —	398.3; 3202.97; —
	468	2	4	4; 8; 0	4; 8; 0	4; 8; 0	No	No; —; —	3600.28; —; —
		4	3	3; 9; 0	**3; *8.8*; 0.2**	**3; *8.8*; 0.2**	0	3; 9[int]; —	357.02; 3244.26; —

[a]Mean number of stations.
[b]Mean number of workers.
[c]Mean smoothness index (without square root).
[int]Integer solution was found within a given CPU time limit.
No: no solution was found within a given CPU time limit.
—: phase was not performed since previous phases had consumed the given CPU time.

instances for which heuristics showed different performance, VNDS produced better solutions than at least one of other two heuristics.

For large size instances, heuristics' results are given in Table 7. For 7 of 18 instances in this group, all three heuristics obtained the same results. For 9 of the remaining 11 instances (82%), RVNS heuristic outperformed at least one of other two heuristics.

5.3. Minimizing Smoothness Index due to Given Target Values of the Number of Stations and Workers. The same method with the one for generating target values for the number of stations was used to obtain target values for the number of workers. The target value for the number of workers for an instance was set to the maximum of 15 values obtained by heuristics for the number of workers in Section 5.1.

The results for small and medium size instances are presented in Table 8. The following observations can be made from this table. By using the MIP formulation, instances with target values may be optimally solved more easily since more instances are solved optimally. Also, in this group, there are only 4 instances for which heuristics showed

Table 7: For a given target value of the number of stations, results for large size instances for minimizing the number of workers, and smoothness index in the same order.

Problem	Ct	M_{max}	Target # stations	VNS result	RVNS result	VNDS result
P83	5048	2	10	10[a]; 16[b]; 4[c]	10; 16; 4	10; 16; 4
		4	9	**9; *17.4*; 13.6**	9; 17.6; 13.4	9; 17.6; 13.4
	6842	2	8	8; 12; 4	8; 12; 4	8; 12; 4
		4	7	7; 13; 3.6	**7; *12.8*; 3.8**	7; 13; 8
	7571	2	7	6.4; 11; 1.8	**6; 11; *1***	6.2; 11; 1.4
		4	6	6; 11; 3	**6; 11; *1***	**6; 11; *1***
P111	5755	2	16	15.8; 28.2; 3.4	**16; *28*; 4**	16; 28.6; 3.4
		4	13	13; 29; 33.2	13; 28.8; 23.2	**13; *28.6*; 39.8**
	8847	2	11	11; 18; 4	11; 18; 4	11; 18; 4
		4	10	10; 18; 36	**10; 18; *20.4***	10; 18; 21.2
	10743	2	9	8.8; 15; 2.6	**8.4; 15; *1.8***	9; 15; 3
		4	7	7; 15.8; 16.8	**7; *15.4*; 25.2**	7; 16; 15.4
P148	434	2	7	7; 14; 0	7; 14; 0	7; 14; 0
		4	5	5; 14; 8	**5; 14; *1***	5; 14; 3.2
	626	2	5	5; 10; 0	5; 10; 0	5; 10; 0
		4	3	3; 10; 2	3; 10; 2	3; 10; 2
	805	2	4	4; 8; 0	4; 8; 0	4; 8; 0
		4	3	2.6; 8; 0.6	**2; 8; *0***	2.4; 8; 0.4

[a]Mean number of stations.
[b]Mean number of workers.
[c]Mean smoothness index (without square root).

different performance. For 3 of these 4 instances (75%), RVNS obtained better solutions than other two heuristics. For the remaining instance, VNDS was superior against other two heuristics.

Similar observations can be made for large size instances whose results are given in Table 9. As is seen from this table, VNS was not able to outperform other heuristics even for a single instance. On the other hand, for 10 instances, at least one of other two heuristics outperformed VNS. For these 10 instances, RVNS obtained the same (for 3 instances) or better results (for 5 instances) than VNDS. For the remaining 2 instances, VNDS outperformed RVNS.

6. Conclusions and Future Works

In this research, multi-objective MALBP is considered. First, a MIP model based on goal programming approach is developed. Then, after defining the problem, three different variations of VNS algorithm are proposed to solve multi-objective MALBP heuristically. Experiment study is performed based on benchmark instances ranging in size from small to large for different target values of problem parameters.

Based on well-known goal programming approach, this study is the first one attempting to define the line balancing problem with multi-manned stations and to propose solution methods to solve it. After analyzing the results of experimental study, it has been seen that only small size instances could be solved optimally. On the other hand, if the instance has target values attainable, then its optimal solution may be found more easily. For small size instances, it has been seen that there is no significant difference between proposed VNS variations. However, for medium and especially large size instances, RVNS mostly outperforms other two heuristics.

A possible future research direction is to develop mathematical programming models for other goal, such as task based smoothness index and cost. Also, to the best of authors' knowledge, there is no study on the MALBP which has layout different from straight layout, such as U line and parallel line. Therefore, there is a need to develop solution algorithms for MALBP having different line layouts.

Competing Interests

The author declares that there are no competing interests.

Table 8: For given target values of the number of stations and workers, results for small and medium size instances for minimizing smoothness index.

Problem	Ct	M_{max}	Target		VNS result	RVNS result	VNDS result	MIP			
			# stations	# workers				d_1^+	d_2^+	Result	CPU
P11	7	2	6	8	6[a]; 8[b]; 4[c]	6; 8; 4	6; 8; 4	0	0	6; 8; 4	0.14; 0.08; 0.33
		4	5	9	5; 9; 10	5; 9; 10	5; 9; 10	0	0	5; 9; 10	0.16; 0.08; 0.09
	10	2	4	5	4; 5; 3	4; 5; 3	4; 5; 3	0	0	4; 5; 3	0.13; 0.11; 0.31
		4	3	6	3; 6; 5	3; 6; 5	3; 6; 5	0	0	3; 6; 5	0.08; 0.08; 0.09
	21	2	2	3	2; 3; 1	2; 3; 1	2; 3; 1	0	0	2; 3; 1	0.11; 0.08; 0.23
		4	2	3	2; 3; 1	2; 3; 1	2; 3; 1	0	0	2; 3; 1	0.09; 0.08; 0.19
P21	14	2	7	8	7; 8; 6	7; 8; 6	7; 8; 6	0	0	7; 8; 6	0.58; 0.27; 0.39
		4	7	8	7; 8; 6	7; 8; 6	7; 8; 6	0	0	7; 8; 6	2.22; 0.83; 0.91
	21	2	4	6	4; 6; 2	4; 6; 2	4; 6; 2	0	0	4; 6; 2	0.92; 0.28; 0.58
		4	4	6	4; 6; 2	4; 6; 2	4; 6; 2	0	0	4; 6; 2	2.62; 0.59; 0.89
	35	2	3	3	3; 3; 0	3; 3; 0	3; 3; 0	0	0	3; 3; 0	0.58; 1.61; 0.17
		4	3	3	3; 3; 0	3; 3; 0	3; 3; 0	0	0	3; 3; 0	2.06; 1.2; 0.28
P30	25	2	8	14	8; 14; 2	8; 14; 2	8; 14; 2	0	0	8; 14; 2	13.03; 0.81; 1.37
		4	8	14	8; 14; 2	8; 14; 2	8; 14; 2	0	0	8; 14; 2	6.02; 4.57; 6.3
	30	2	7	12	7; 12; 2	**6.8; 12; *1.6***	7; 12; 2	0	0	7; 12; 0	7.55; 1.29; 8.18
		4	6	12	6; 12; 8	6; 12; 10	**6; 12; *6***	0	0	6; 12; 0	15.12; 1.3; 62.39
	41	2	5	9	5; 9; 1	5; 9; 1	5; 9; 1	0	0	5; 8; 0	11.73; 1.36; 142.48
		4	4	9	4; 9; 3	**4; 8.8; *2.4***	4; 9; 3	0	0	4; 8; 0	10.17; 2.96; 2977.37
P45	57	2	6	10	6; 10; 2	6; 10; 2	6; 10; 2	0	No	6; No; —	253.24; 3346.86; —
		4	5	10	5; 10; 7	5; 10; 7	5; 10; 7	0	0	5;10;No	73.68; 106.18; 3420.32
	110	2	3	6	3; 6; 0	3; 6; 0	3; 6; 0	0	0	3; 6; 0	53.95; 2.28; 2.23
		4	3	6	3; 6; 0	3; 6; 0	3; 6; 0	0	0	3; 6; 0	95.22; 7.64; 23.46
	184	2	2	3	2; 3; 1	2; 3; 1	2; 3; 1	0	0	2; 3; No	39.22; 527.08; 3033.8
		4	2	3	2; 3; 1	2; 3; 1	2; 3; 1	0	0	2; 3; 1[int]	60.23; 2601.69; 938.55
P70	176	2	12	22	12; 22; 2	12; 22; 2	12; 22; 2	No	—	No; —; —	3600.27; —; —
		4	9	23	9; 23; 6	**9; 23; *5.2***	9; 23; 9.4	0	0	9; 23; No	496.6; 121.28; 2982.48
	364	2	6	10	6; 10; 2	6; 10; 2	6; 10; 2	No	—	No; —; —	3600.27; —; —
		4	4	12	4; 12; 0	4; 12; 0	4; 12; 0	0	0	4; 12; 0	398.05; 66.35; 75.41
	468	2	4	8	4; 8; 0	4; 8; 0	4; 8; 0	No	—	No; —; —	3600.3; —; —
		4	3	9	3; 9; 0	3; 9; 0	3; 9; 0	0	0	3; 9; 0	355.39; 62.01; 137.83

[a]Mean number of stations.
[b]Mean number of workers.
[c]Mean smoothness index (without square root).
[int]Integer solution was found within a given CPU time limit.
No: no solution was found within a given CPU time limit.
—: phase was not performed since previous phases had consumed the given CPU time.

Table 9: For given target values of the number of stations and workers, results for large size instances for minimizing smoothness index.

Problem	Ct	M_{max}	Target		VNS result	RVNS result	VNDS result
			# stations	# workers			
P83	5048	2	10	16	10[a]; 16.2[b]; 3.8[c]	**10; *16*; 4**	**10; *16*; 4**
		4	9	18	9; 18; 13	9; 18; 12.6	**9; 18; *12.2***
	6842	2	8	12	8; 12; 4	8; 12; 4	8; 12; 4
		4	7	13	7; 13; 7.6	**7; 13; *1***	7; 13; 7.6
	7571	2	7	11	6.6; 11; 2.2	**6.4; 11; *1.8***	**6.4; 11; *1.8***
		4	6	11	6; 11; 1	6; 11; 1	6; 11; 1
P111	5755	2	16	29	16; 29; 3	16; 29; 3	**15.8; 29; *2.6***
		4	13	29	13; 29; 15.6	**13; 29; *14.8***	13; 29; 23
	8847	2	11	18	11; 18; 4	11; 18; 4	11; 18; 4
		4	10	18	10; 18; 21.2	**10; 18; *20.4***	10; 18; 28.4
	10743	2	9	15	8.4; 15; 1.8	**8.2; 15; *1.4***	8.8; 15; 2.6
		4	7	17	7; 17; 9.4	**7; 17; *5.6***	7; 17; 7.8
P148	434	2	7	14	7; 14; 0	7; 14; 0	7; 14; 0
		4	5	14	5; 14; 11.4	**5; 14; *1***	**5; 14; *1***
	626	2	5	10	5; 10; 0	5; 10; 0	5; 10; 0
		4	3	10	3; 10; 2	3; 10; 2	3; 10; 2
	805	2	4	8	4; 8; 0	4; 8; 0	4; 8; 0
		4	3	8	3; 8; 1	**2; 8; *0***	2.2; 8; 0.2

[a]Mean number of stations.
[b]Mean number of workers.
[c]Mean smoothness index (without square root).

References

[1] E. Gurevsky, Ö. Hazır, O. Battaïa, and A. Dolgui, "Robust balancing of straight assembly lines with interval task times," *Journal of the Operational Research Society*, vol. 64, no. 11, pp. 1607–1613, 2013.

[2] H. Gökcen and E. Erel, "Binary integer formulation for mixed-model assembly line balancing problem," *Computers and Industrial Engineering*, vol. 34, no. 2–4, pp. 451–461, 1998.

[3] P. Sivasankaran and P. Shahabudeen, "Literature review of assembly line balancing problems," *International Journal of Advanced Manufacturing Technology*, vol. 73, no. 9–12, pp. 1665–1694, 2014.

[4] A. Lusa, "A survey of the literature on the multiple or parallel assembly line balancing problem," *European Journal of Industrial Engineering*, vol. 2, no. 1, pp. 50–72, 2008.

[5] C. Becker and A. Scholl, "A survey on problems and methods in generalized assembly line balancing," *European Journal of Operational Research*, vol. 168, no. 3, pp. 694–715, 2006.

[6] E. Erel and S. C. Sarin, "A survey of the assembly line balancing procedures," *Production Planning and Control*, vol. 9, no. 5, pp. 414–434, 1998.

[7] C. Becker and A. Scholl, "Balancing assembly lines with variable parallel workplaces: problem definition and effective solution procedure," *European Journal of Operational Research*, vol. 199, no. 2, pp. 359–374, 2009.

[8] T. Kellegöz and B. Toklu, "An efficient branch and bound algorithm for assembly line balancing problems with parallel multi-manned workstations," *Computers and Operations Research*, vol. 39, no. 12, pp. 3344–3360, 2012.

[9] S. G. Dimitriadis, "Assembly line balancing and group working: a heuristic procedure for workers' groups operating on the same product and workstation," *Computers and Operations Research*, vol. 33, no. 9, pp. 2757–2774, 2006.

[10] E. Cevikcan, M. B. Durmusoglu, and M. E. Unal, "A team-oriented design methodology for mixed model assembly systems," *Computers and Industrial Engineering*, vol. 56, no. 2, pp. 576–599, 2009.

[11] P. Fattahi, A. Roshani, and A. Roshani, "A mathematical model and ant colony algorithm for multi-manned assembly line balancing problem," *International Journal of Advanced Manufacturing Technology*, vol. 53, no. 1–4, pp. 363–378, 2011.

[12] V. Yazdanparast and H. Hajihosseini, "Multi-manned production lines with labor concentration," *Australian Journal of Basic and Applied Sciences*, vol. 5, no. 6, pp. 839–846, 2011.

[13] A. Kazemi 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, no. 3, pp. 289–309, 2013.

[14] T. Kellegöz and B. Toklu, "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, pp. 736–756, 2015.

[15] A. Scholl, *Balancing and Sequencing of Assembly Lines*, Physica-Verlag, Heidelberg, Germany, 1999.

[16] A. Charnes and W. W. Cooper, *Management Model and Industrial Application of Linear Programming*, John Wiley & Sons, New York, NY, USA, 1961.

[17] W. L. Winston, *Operations Research: Applications and Algorithms*, Duxbury Press, Belmont, Calif, USA, 3rd edition, 1994.

[18] H. A. Taha, *Operations Research: An Introduction*, Pearson, New Jersey, NJ, USA, 8th edition, 2007.

[19] Y. Wilamowsky, S. Epstein, and B. Dickman, "Optimization in multiple-objective linear programming problems with pre-emptive priorities," *Journal of the Operational Research Society*, vol. 41, no. 4, pp. 351–356, 1990.

[20] N. Mladenović, "A variable neighborhood algorithm—a new metaheuristic for combinatorial optimization," in *Abstracts of Papers Presented at Optimization Days*, p. 112, Montreal, Canada, 1995.

[21] P. Hansen and N. Mladenović, "Variable neighborhood search: principles and applications," *European Journal of Operational Research*, vol. 130, no. 3, pp. 449–467, 2001.

[22] P. Hansen, N. Mladenović, and D. Perez-Britos, "Variable neighborhood decomposition search," *Journal of Heuristics*, vol. 7, no. 4, pp. 335–350, 2001.

[23] P. Hansen, N. Mladenović, and J. A. M. Pérez, "Variable neighbourhood search: methods and applications," *4OR*, vol. 6, no. 4, pp. 319–360, 2008.

[24] R. K. Hwang, H. Katayama, and M. Gen, "U-shaped assembly line balancing problem with genetic algorithm," *International Journal of Production Research*, vol. 46, no. 16, pp. 4637–4649, 2008.

[25] U. Özcan and B. Toklu, "Multiple-criteria decision-making in two-sided assembly line balancing: a goal programming and a fuzzy goal programming models," *Computers & Operations Research*, vol. 36, no. 6, pp. 1955–1965, 2009.

[26] M. L. Pinedo, *Scheduling Theory, Algorithms and Systems*, Prentice Hall, New Jersey, NJ, USA, 2008.

[27] P. Hansen and N. Mladenović, "An Introduction to variable neighborhood search," in *Metaheuristics, Advances and Trends in Local Search Paradigms for Optimization*, S. Voss, S. Martello, I. H. Osman, and C. Roucairol, Eds., pp. 433–458, Kluwer Academic, Dordrecht, The Netherlands, 1998.

[28] J. R. Jackson, "A computing procedure for a line balancing problem," *Management Science*, vol. 2, no. 3, pp. 261–271, 1956.

[29] J. Mitchell, "A computational procedure for balancing zoned assembly lines," Research Report 6-94801-1-R3, Westinghouse Research Laboratories, Pittsburgh, Pa, USA, 1957.

[30] J. F. H. Sawyer, *Line Balancing*, Machinery and Allied Products Institute, Washington, DC, USA, 1970.

[31] M. D. Kilbridge and L. Wester, "A heuristic method of assembly line balancing," *Journal of Industrial Engineering*, vol. 12, pp. 292–298, 1961.

[32] F. M. Tonge, *A Heuristic Program of Assembly Line Balancing*, Prentice Hall, Englewood Cliffs, NJ, USA, 1961.

[33] A. L. Arcus, *An analysis of a computer method of sequencing assembly line operations [Ph.D. dissertation]*, University of California, 1963.

[34] J. J. Bartholdi, "Balancing two–sided assembly lines: a case study," *International Journal of Production Research*, vol. 31, no. 10, pp. 2447–2461, 1993.