DOI: 10.3901/CJME.2015.0630.084, available online at www.springerlink.com; www.cjmenet.com; www.cjme.com.cn

Effective Hybrid Teaching-learning-based Optimization Algorithm for Balancing Two-sided Assembly Lines with Multiple Constraints

TANG Qiuhua[1, *], LI Zixiang[1], ZHANG Liping[1], FLOUDAS C A[2], and CAO Xiaojun[3]

1 *Industrial Engineering Department, Wuhan University of Science and Technology, Wuhan 430081, China*
2 *Texas A&M Energy Institute, Texas A&M University, College Station TX 77843, USA*
3 *Technique Center of Dongfeng Peugeot Citroen Automobile Company, Wuhan 430056, China*

Received January 26, 2014; revised March 5, 2015; accepted June 30, 2015

Abstract: Due to the NP-hardness of the two-sided assembly line balancing (TALB) problem, multiple constraints existing in real applications are less studied, especially when one task is involved with several constraints. In this paper, an effective hybrid algorithm is proposed to address the TALB problem with multiple constraints (TALB-MC). Considering the discrete attribute of TALB-MC and the continuous attribute of the standard teaching-learning-based optimization (TLBO) algorithm, the random-keys method is hired in task permutation representation, for the purpose of bridging the gap between them. Subsequently, a special mechanism for handling multiple constraints is developed. In the mechanism, the directions constraint of each task is ensured by the direction check and adjustment. The zoning constraints and the synchronism constraints are satisfied by teasing out the hidden correlations among constraints. The positional constraint is allowed to be violated to some extent in decoding and punished in cost function. Finally, with the TLBO seeking for the global optimum, the variable neighborhood search (VNS) is further hybridized to extend the local search space. The experimental results show that the proposed hybrid algorithm outperforms the late acceptance hill-climbing algorithm (LAHC) for TALB-MC in most cases, especially for large-size problems with multiple constraints, and demonstrates well balance between the exploration and the exploitation. This research proposes an effective and efficient algorithm for solving TALB-MC problem by hybridizing the TLBO and VNS.

Keywords: two-sided assembly line balancing, teaching-learning-based optimization algorithm, variable neighborhood search, positional constraints, zoning constraints, synchronism constraints

1 Introduction

Two-sided assembly lines usually exist in plants which produce large-sized high-volume products, such as buses and trucks. The two-sided assembly line has many advantages over the well-known one-sided assembly line, including (1) shorter line length, (2) less throughput and setup times, (3) less cost of fixtures and tools, and (4) less material handling[1]. On two-sided assembly lines, different tasks of each product can be operated in parallel on both sides. Some tasks must be operated on the left or right side (called L or R), while others can be performed on either side (called E). A pair of directly facing stations composes a mated-position in two-sided assembly line[2]. As shown in Fig. 1, there are 8 stations and 4 mated-stations.

The assembly line balancing (ALB) problem is allocating the tasks to an ordered sequence of stations. The main difference between the two-sided and one-sided assembly line balancing problem is the restriction on the operation directions. Let's consider the 9-task problem introduced by KIM, et al[3], in Fig. 2.

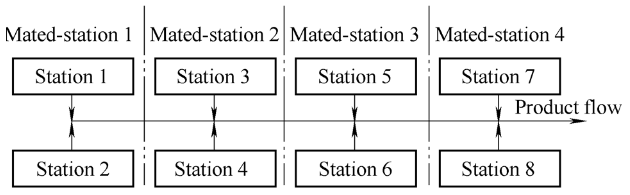

Fig. 1. A two-sided assembly line with four mated-stations

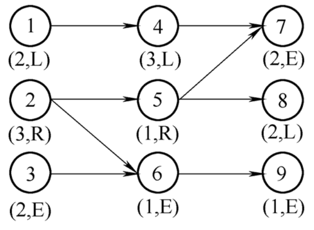

Fig. 2. Precedence graph of the 9-task problem

The operation times and the operation directions (R, L or E) are shown on the precedence diagram for all the tasks. The tasks are represented by the numbers in the nodes. The

* Corresponding author. E-mail: tangqiuhua@wust.edu.cn

Supported by National Natural Science Foundation of China (Grant Nos. 51275366, 50875190, 51305311) and Specialized Research Fund for the Doctoral Program of Higher Education of China (Grant No. 20134219110002)

labels (t_i, d_i) below the nodes represent the task times (t_i) and the preferred operation directions (d_i), e.g., the task time of task 1 is 2 and the corresponding preferred operation direction is L. The precedence relations among tasks are represented with the directed arrows between two task nodes, e.g., task 2 is the immediate predecessor of task 5.

However, in some practical applications, there are three additional constraints needed to be taken into account, including the positional constraints, zoning constraints and synchronism constraints. (1) Positional constraint means that certain tasks should be assigned to a predetermined station[3]. For example, whenever large, heavy or immovable facilities are installed, the locations must be fixed and the tasks using these facilities must be assigned to these fixed stations. (2) The zoning constraints are about the closeness preference of tasks. There are two kinds of zoning constraints, namely positive zoning constraint and negative zoning constraint[4]. Positive zoning constraint indicates that a set of tasks must be allocated to the same station. Negative zoning constraint means that a set of tasks are prohibited to be allocated to the same station or mated-station. For instance, if a set of tasks need different equipment, so they can’t be operated at the same station. (3) The synchronism constraint exists when a pair of tasks needs to be operated simultaneously on both sides of the line, so that two operators within the same mated-station can collaborate[5]. All these additional constraints make it even more difficult to solve the two-sided assembly line balancing (TALB) problem.

This paper focuses on solving the two-sided assembly line balancing problem with multiple constraints (TALB-MC), with the objective of minimizing the number of stations and mated-stations simultaneously. A hybrid algorithm combining teaching-learning-based optimization (TLBO) and variable neighborhood search (VNS) is developed to solve this problem. TLBO is first proposed by RAO, et al[6] and it outperforms some of the well-known meta-heuristics regarding constrained benchmark functions, constrained mechanical design, and continuous non-linear numerical optimization problems[7–8]. It is considered as an algorithm-specific parameter-less algorithm and the advantages of TLBO algorithm such as speediness and robustness are shown in the literature. What’s more, VNS is a recent meta-heuristic algorithm based on the principle of systematic change of the neighborhood during the search process and two or more neighborhoods are used in its structure[9]. Thus, we employ VNS algorithm as a robust local search-based approach to improve the solutions generated by TLBO. In this algorithm, TLBO is utilized for global search and the VNS is used to intensify the solutions to find better local solutions, making this algorithm achieve the proper balance between intensification and diversification. This paper is the first attempt to use the hybrid TLBO for TALB-MC problem.

The organization of this paper is presented as follows. The literature review is given below after the introduction in section 2. Then, the mathematical formulation for TALB-MC problem is introduced in section 3. In section 4, the hybrid TLBO algorithm for solving TALB-MC problem is described in details. Then computational experiments and comparison results are carried out to demonstrate the performance of the proposed algorithm. At last, the conclusions are provided in section 6.

2 Literature Review

As far as we know, BARTHOLDI[1] was the first researcher to study TALB problems. In his study, an assignment rule was employed to assign tasks. LEE, et al[2], developed an assignment procedure to maximize relatedness and work slackness.KIM, et al[3], presented a genetic algorithm (GA) for TALB problems with the objective of minimizing the number of stations. HU, et al[10], presented a station-oriented enumerative algorithm and provided the lower bound of the number of stations. KIM, et al[11], constructed a mathematical model with the objective of minimizing the cycle time for a given number of stations and employed a genetic algorithm. WU, et al[12] and HU, et al[13], proposed the branch-and-bound algorithms to solve the standard TALB problem. CHUTIMA, et al[14], developed the particle swarm optimization (PSO) algorithm with negative knowledge and solved the mixed-model TALB problem with four objectives.

For TALB real-life problem, ÖZCAN[15] built a chance-constrained, piecewise-linear, mixed integer programming model for the stochastic TALB problem. ÖZBAKIR, et al[16], used the bee algorithm (BA) to balance the fuzzy multi-objective two-sided assembly lines to handle the imprecise objectives. Moreover, different kinds of constraints existing in practice are considered gradually. BAYKASOGLU, et al[4], developed an ant-colony-based heuristic for two-sided assembly line balancing with zoning constraints. SIMARIA, et al[5], used the ant colony optimization (ACO) algorithm to solve the mixed-model TALB problem with the zoning constraints and the synchronism constraints. ÖZCAN, et al[17–19], developed a tabu search (TS) algorithm to maximize the line efficiency and minimize the smoothness index, proposed the goal programming models with multiple objectives and the zoning constraints, then extended the mathematical model of TALB with consideration of multiple constraints, including the positional constraints,zoning constraints and synchronism constraints in the same year. ÖZBAKIR, et al[20], used the bee algorithm (BA) to solve the TALB problem with the zoning constraints. TAPKAN, et al[21], proposed the bee algorithm (BA) to solve the fuzzy multi-objective two-sided assembly line balancing problem with the three additional constraints. BIAO, et al [22], solved two-sided assembly lines with multiple constraints using late acceptance hill-climbing algorithm.

However, there is little published paper dealing with TALB problem using TLBO. In addition, little attention has been paid to solve the TALB problem with the positional constraints, zoning constraints and synchronism constraints. Therefore, this study focuses on solving the TALB problem with all the three additional constraints using hybrid TLBO.

3 Mathematical Formulation

3.1 Problem assumptions

The assumptions for the TALB-MC problem in this study are introduced as follows.

(1) A single-model is considered. TALB problem with several models is also applicable via combined precedence relations[21].

(2) The processing times for all tasks are deterministic and fixed.

(3) The tasks are operated in parallel on both sides of the line within a given cycle time.

(4) No buffer is involved.

(5) A task can be operated only if all its predecessors have been completed.

(6) Each task must be assigned to only one station.

(7) Tasks with the positive zoning constraints must be operated at the same station.

(8) Every task in the positive zoning constraints should follow each other and their immediate predecessors must be members of the corresponding set, except the first preceding task[4].

(9) Tasks with the synchronism constraints must be operated concurrently on both sides of the same mated-station.

(10) A pair of tasks with the negative zoning constraints can't be assigned to the same mated-station[4].

3.2 Notations

The notations used in the mathematical formulation are listed as follows and other notations will be described if necessary.

(1) Parameters:

nt —Number of tasks,

I —Set of tasks, $I = \{1, 2, \cdots, i, \cdots, nt\}$,

nm —Number of mated-stations,

J —Set of mated-stations, $J = \{1, 2, \cdots, j, \cdots, nm\}$,

k —Indicator for the side of the line. $k = 1$, if the side is left; $k = 2$, otherwise,

(j, k) —Station of mated-station j and direction k.

AL —Set of tasks that should be performed at the left station; $AL \subseteq I$,

AR —Set of tasks that should be performed at the right station; $AR \subseteq I$,

AE —Set of tasks that can be performed on either side of the mated-station; $AE \subseteq I$,

P_0 —Set of tasks that have no immediate predecessors,

$P_a(i)$ —Set of predecessors of the task i,

$P(i)$ —Set of immediate predecessors of the task i,

$S(i)$ —Set of immediate successors of the task i,

$S_a(i)$ —Set of successors of the task i,

$C(i)$ —Set of tasks whose operation direction is opposite to task i's operation direction.

$K(i)$ —Set of integers which indicate the assignable operation directions of the task i,

$$K(i) = \begin{cases} \{1\}, & \text{if } i \in AL, \\ \{2\}, & \text{if } i \in AR, \\ \{1, 2\}, & \text{if } i \in AE, \end{cases}$$

PZ —Set of pairs of tasks for positive zoning constraints,

NZ —Set of pairs of tasks for negative zoning constraints,

PC —Set of pairs of tasks and predetermined station for positional constraints,

SC —Set of pair of tasks for synchronism constraints,

t_i —Processing time of task i,

CT —Cycle time,

μ —Large positive number,

ns —Number of stations,

w_{nm} —The weighted cost for opening one mated-station.

w_{ns} —The weighted cost for opening one station.

(2) Decision variables:

t_i^s —Positive variable, the start time of task i,

$$x_{ijk} = \begin{cases} 1, & \text{if task } i \text{ is assigned to station } (j, k), \\ 0, & \text{otherwise,} \end{cases}$$

$$F_j = \begin{cases} 1, & \text{if both sides of mated-station } j \text{ are utilized,} \\ 0, & \text{otherwise,} \end{cases}$$

$$G_j = \begin{cases} 1, & \text{if only one side of mated-station } j \text{ is utilized,} \\ 0, & \text{otherwise,} \end{cases}$$

$$U_{jk} = \begin{cases} 1, & \text{if station } (j, k) \text{ is utilized,} \\ 0, & \text{otherwise.} \end{cases}$$

(3) Indicator variables:

$$z_{ip} = \begin{cases} 1, & \text{if task } i \text{ is assigned earlier than task } p \text{ in the same station,} \\ 0, & \text{otherwise.} \end{cases}$$

3.3 Mathematical model

The objective of the problem is minimizing the number of mated-stations and the number of stations simultaneously within a given cycle time. Since it is a multi-objective problem, we use the linear weighting method to combine the two objectives into one.

$$\min w_{nm} \cdot \sum_{j \in J} (F_j + G_j) + w_{ns} \cdot \sum_{j \in J} \sum_{k=1,2} U_{jk}, \tag{1}$$

s.t.,

(1) Occurrence constraint. Any task is assigned to only one station:

$$\sum_{j \in J} \sum_{k \in K(i)} x_{ijk} = 1, \text{ for } \forall i \in I. \tag{2}$$

(2) Precedence constraint:

$$\sum_{g\in J}\sum_{k\in K(h)} g \cdot x_{hgk} \leqslant \sum_{j\in J}\sum_{k\in K(i)} j \cdot x_{ijk}, \text{ for } \forall i \in I - P_0,\, h \in P(i), \tag{3}$$

$$t_i^s - t_h^s + \mu(1 - \sum_{k\in K(h)} x_{hjk}) + \mu(1 - \sum_{k\in K(i)} x_{ijk}) \geqslant t_h, \quad \text{for } \forall i \in I - P_0,\, h \in P(i),\, j \in J, \tag{4}$$

$$t_p^s - t_i^s + \mu(1 - x_{ijk}) + \mu(1 - x_{pjk}) + \mu(1 - z_{ip}) \geqslant t_i, \quad \text{for } i \in I, j \in J, k \in K(i) \cap K(p), \quad p \in \{r \mid r \in I - (P_a(i) \cup S_a(i) \cup C(i)) \text{ and } i < r\}, \tag{5}$$

$$t_i^s - t_p^s + \mu(1 - x_{ijk}) + \mu(1 - x_{pjk}) + \mu \cdot z_{ip} \geqslant t_p, \quad \text{for } i \in I, j \in J, k \in K(i) \cap K(p), \quad p \in \{r \mid r \in I - (P_a(i) \cup S_a(i) \cup C(i)) \text{ and } i < r\}. \tag{6}$$

Eq. (3) is the precedence constraint. If *h* is the immediate predecessor of *i* and *i* is assigned to mated-station *j*, then task *h* must be assigned to the mated-station between 1 and *j*. However, Eq. (3) is not sufficient for the pair of tasks assigned to the same mated-station and thus Eqs. (4)–(6) are used. For each pair of tasks (*i*, *h*), if task *h* is an immediate processor of task *i* and they are assigned to the same mated-station, then Eq. (4) becomes active and it is reduced to $t_i^s - t_h^s \geqslant t_h$. For each pair of tasks (*i*, *p*), if there are no precedence relations between them and they are at the same station (*j*, *k*), then Eqs. (5) and (6) become active. If task *i* is assigned earlier than task *p*, then $z_{ip} = 1$; otherwise, $z_{ip} = 0$. If task *i* is assigned earlier than task *p* at the same station, Eq. (5) becomes active and it is reduced to $t_p^s - t_i^s \geqslant t_i$; otherwise, Eq. (6) becomes active and it is reduced to $t_i^s - t_p^s \geqslant t_p$.

(3) Cycle time constraint:

$$t_i^s + t_i \leqslant CT, \text{ for } \forall i \in I. \tag{7}$$

(4) Positional constraint. Tasks with the positional constraint should be assigned to the preferred mated-station and the preferred direction:

$$x_{ijk} = 1, \text{for} \forall (i, (j, k)) \in PC. \tag{8}$$

(5) Zoning constraint. Tasks with the positive zoning constraint should be assigned to the same mated-station and the same side. And tasks with the negative zoning constraints shouldn't be assigned to the same mated-station.

Positive zoning:

$$x_{ijk} - x_{hjk} = 0, \text{ for } (i, h) \in PZ. \tag{9}$$

Negative zoning:

$$\sum_{k\in K(i)} x_{ijk} + \sum_{k\in K(h)} x_{hjk} \leqslant 1, \text{ for } (i, h) \in NZ. \tag{10}$$

(6) Synchronism constraint. A pair of tasks with synchronism constraint is operated at different side of the same mated-station and they have the same start time:

$$x_{ijf} - x_{hjk} = 0, \text{ for } (i, h) \in SC, k \neq f, \tag{11}$$

$$t_i^s - t_h^s = 0, \text{ for } (i, h) \in SC. \tag{12}$$

(7) Station constraint and mated-station constraint.

Station constraint:

$$\sum_{i\in I} x_{ijk} - \mu \cdot U_{jk} \leqslant 0, \text{for} \forall j \in J, k \in K(i). \tag{13}$$

Eq. (13) ensures that if station (*j*, *k*) is used, then $U_{jk} = 1$; else, $U_{jk} = 0$.

Mated-station constraint:

$$\sum_{k=1,2} U_{jk} - 2 \cdot F_j - G_j = 0, \text{ for} \forall j \in J. \tag{14}$$

Eq. (14) ensures that if mated-station *j* is used for both sides, $F_j = 1$; $F_j = 0$, otherwise. If mated-station *j* is used for only side, $G_j = 1$; $G_j = 0$, otherwise. Therefore, the number of stations is equal to the sum of U_{jk} and the number of mated-stations is equal to the sum of F_j and G_j. Eqs. (15)–(19) are the integrality and non-negativity constraints:

$$x_{ijk} \in \{0, 1\}, \quad \text{for} \forall i \in I, j \in J, k \in K(i), \tag{15}$$

$$z_{ip} \in \{0,1\}, \text{ for} \forall i \in I, \quad p \in \{r \mid r \in I - (P_a(i) \cup S_a(i) \cup C(i)) \text{ and } i < r\}, \tag{16}$$

$$U_{jk} \in \{0, 1\}, \quad \text{for} \forall j \in J, k = 1, 2, \tag{17}$$

$$F_j, G_j \in \{0, 1\}, \quad \text{for} \forall j \in J, \tag{18}$$

$$t_i^s \geqslant 0, \quad \text{for } \forall i \in I. \tag{19}$$

4 Hybrid TLBO Algorithm for TALB Problems

TALB problems are difficult to be solved with exact algorithms within limited CPU time due to its large-size feature, thus, heuristics or meta-heuristics are adopted. In this paper, a hybrid algorithm combining teaching-learning-based optimization (TLBO) and variable neighborhood search (VNS) is proposed. Where, the TLBO is used for global search, and the VNS is used to intensify the solutions to find better local solutions. In this way, the proper balance between intensification and diversification can be achieved. The main procedure of the hybrid TLBO algorithm for TALB-MC is described in the following subsections.

4.1 TLBO algorithm

TLBO is first published in Ref. [6]. TLBO needs only common controlling parameters like population size and number of the iterations, therefore, it can be considered as an algorithm-specific parameter-less algorithm. What's more, TLBO algorithm requires less iteration for convergence to the optimal solution compared with other algorithms[8].

TLBO is a population-based method and it uses a population to proceed to the optimal solution. The main idea behind TLBO is the simulation of students' learning process in class. It is inspired by the effect of the learning through a teacher and interacting with the other learner in a class. A class of students is regarded as the population and the best solution is regarded as the teacher. It consists of two stages: the teacher phase and learner phase.

During the teacher phase, the students learn from the teacher and obtain the knowledge. During Learner Phase, a student may learn and improve himself or herself through peer learning amongst fellow students. Just like other population-based algorithm, the TLBO algorithm has a predefined size of the population (*pop_size*). A single possible solution is represented by a student Y_i for a particular optimization problem. Y_i is a real-value vector with D elements, D is the dimension of the problem. The goal of this algorithm is to improve individuals by the teacher and learner phases. The current solution will be replaced only when his/her new solution is better than his/her original one. The algorithm will end when it reaches the maximum number of generations.

During the teacher phase, the current best individual is selected as the teacher (Y_{teacher}). Other individuals (Y_i) will be improved based on the Y_{teacher} and the current mean value of all individuals (Y_{mean}). The teacher tries to improve the mean result of the classroom from the Y_{mean} to his or her level (Y_{teacher}). In fact, it is impossible, but a teacher can move the mean of a classroom Y_{mean} to a better value $Y_{\text{mean}}^{\text{new}}$. The difference between Y_{mean} and $Y_{\text{mean}}^{\text{new}}$ is given by Eq. (20):

$$Difference\_Mean = r(Y_{\text{mean}}^{\text{new}} - T_F Y_{\text{mean}}), \tag{20}$$

where the parameter r ranges between 0 and 1 and T_F is teaching factor which decides the value of mean to be changed. The value of T_F can be either 1 or 2 which is a heuristic step and it is decided randomly with equal probability as

$$T_F = round\,(1 + \text{rand}(0,1)). \tag{21}$$

Based on this $Difference\_Mean$, the existing solution Y_i is updated by Eq. (22):

$$Y_i^{\text{new}} = Y_i + Difference\_Mean. \tag{22}$$

The Y_{teacher} tries to improve the existing mean Y_{mean} towards himself or herself, so the new mean is designated as Y_{teacher} ($Y_{\text{mean}}^{\text{new}} = Y_{\text{teacher}}$). Therefore, Eq. (22) can be replaced by Eq. (23). Eq. (23) simulates how student improvement may be influenced by the teacher's knowledge and the qualities of all students:

$$Y_i^{\text{new}} = Y_i + r(Y_{\text{teacher}} - T_F Y_{\text{mean}}). \tag{23}$$

During the learner phase, student (Y_i) tries to improve his/her knowledge by peer learning from an arbitrary student Y_{ii}, where i is unequal to ii. In the case of that student (Y_i) is better than Y_{ii}, Y_i moves away from Y_{ii}. Otherwise, it moves towards Y_{ii}. If Y_i^{new} gets better function value than Y_i by calculating with Eq. (24), Y_i is replaced by Y_i^{new}:

$$Y_i^{\text{new}} = \begin{cases} Y_i + r \bullet (Y_i - Y_{ii}), & Y_i \text{ is better than } Y_{ii}, \\ Y_i + r \bullet (Y_{ii} - Y_i), & \text{otherwise.} \end{cases} \tag{24}$$

Since TALB problems are discrete combination optimization problem in nature, the standard encoding scheme of the TLBO is improper to be implemented directly. For example, floating-point vectors and specific encoding schemes were utilized to represent permutation in BEAN[23]. In this paper, random-keys method similar to BEAN[23] is used to solve the TALB-MC problem. In the next subsection, we explain how this method works.

4.2 Encoding scheme with random-keys method

TLBO was initially designed to solve continuous optimization problems and the standard TLBO could not be employed to solve TALB-MC directly. Thus, random-keys method is used to represent permutation.

In this subsection, we explain the random-keys method by considering TALB-MC problems with 9 tasks in Fig. 2 as an example. First we randomly generated 9 floating-point numbers between 0 and 1, and suppose the vector ψ =(0.42, 0.68, 0.35, 0.01, 0.70, 0.25, 0.79, 0.59, 0.63). We must specify: the position with the lowest value should be first in the task permutation. Thus, the position 4 with the lowest value 0.01 should be first and then the position 6 with the second lowest value 0.25 should be the second in the task permutation. Eventually we can get the task permutation S (4, 6, 3, 1, 8, 9, 2, 5, 7) by repeating this procedure for all numbers. This sequence demonstrates that: the former task in the task permutation has higher priority in task assignment and should be selected at first, e.g., first task 4 is assigned, then task 6, after that task 3, and so on.

4.3 Decoding scheme

The task permutation generated by random-keys method is not a feasible solution, so we develop a procedure based on BIAO, et al[22] to achieve a feasible solution. To explain this decoding scheme clearly, the example in Fig. 2 is applied once again and the involved multiple constraints are described in Table 1.

Table 1. Additional constraints for 9-task problem

Constraint	Positional constraints	Zoning constraints		Synchronism constraint
		Positive	Negative	
Description	{task, (mated-station, direction)}	{task, task}		{task, task}
P9	{1, (1, 1)}, {6, (2, 2)}	{5, 6}	{5, 7}	{4, 5}

4.3.1 *Direction check and adjustment*

We use the direction check to make sure that the additional constraints related to directions are reasonable. The following conditions about the direction check are necessary.

(1) The directions of all tasks in the same positive zoning set should not be L and R simultaneously.

(2) The directions of two tasks in the synchronism set cannot only be L or R.

(3) The direction of the task in positional set should be in accordance with the positional constraint. For example, the direction of task 1 cannot be R because task 1 must be assigned to the left side of the mated-station 1.

If the above necessary conditions have been fully satisfied, the following direction adjustment can be employed to reduce the search space.

(1) Adjust the directions of all tasks in the positive zoning set. If the directions include L and E, change E into L; if R and E, then change E into R.

(2) Adjust the directions of the tasks in the synchronism set. If the directions include L and E, change E into R; if R and E, then change E into L.

(3) Adjust the directions of the tasks in positional set. If the direction of the positional constraint is L or R and the direction of the task with positional constraint is E, change E into L or R.

The direction adjustment can reduce the computational effort and help in finding a feasible solution. The direction adjustment for the test instance is shown in Table 2.

Table 2. Direction adjustment for 9-task problem

Task	1	2	3	4	5	6	7	8	9
Original direction	L	R	E	L	R	***E***	E	L	E
Direction adjustment	L	R	E	L	R	***R***	E	L	E

4.3.2 *Handle the tasks with multiple constraints*

BIAO, et al[22] proposed a decoding scheme for TALB-MC, but the tasks with several additional constraints are not addressed. For example, task 5 in Table 2 is in both positive zoning constraints and synchronism constraint. Therefore, we propose the following method to handle all the possible combinations with two of the three additional constraints in Table 3.

Table 3. Constraint adjustment for 9-task problem

Constraint	Positional constraints	Zoning constraints		Synchronism constraint	Positive-synchronism constraints
		Positive	Negative		
Description	{task, (mated-station, direction)}	{task, task}	{task, task}	{task, task}	{(task, task), (task, task)}
Before handling	{1, (1, 1)}, {6, (2, 2)}	{5, 6}	{5, 7}	{4, 5}	–
After handling	{1, (1, 1)}, {6, (2, 2)}, {5, (2, 2)}, {4, (2, 1)}	{5, 6}	{5, 7}, {6, 7}, {4, 7}	{4, 5}	{(5, 6), (4, 5)}

(1) Tasks with positive zoning constraint and negative constraint.

If a task with the positive zoning constraint is also in the negative constraint, we set that other tasks with the positive zoning constraint have the same negative constraint. For example, {5, 6} is in positive zoning constraint, {5, 7} is in negative zoning constraint, another negative constraint {6, 7} can be appended.

(2) Tasks with negative zoning constraint and synchronism constraint

If a task with the synchronism constraint is also in the negative constraint, we set that the other task also has the negative zoning constraint, e.g., {4, 7}.

(3) Tasks with positional constraint and positive zoning constraint.

We just set that all the other tasks with this positive zoning constraint have the same positional constraint, e.g., {5, (2, 2)}.

(4) Tasks with positional constraint and negative zoning constraint.

In our decoding scheme, the negative zoning constraint should be met and the positional constraints are allowed to be violated. It suggests that a solution may be infeasible. However, the infeasibility of the solution due to the positional constraints may result in huge cost as shown in Eq. (25).

(5) Tasks with positional constraint and synchronism constraint.

If one task has the synchronism and positional constraint simultaneously, the other task in the synchronism set must be assigned to the other side of the mated-station. For example, because of {4, 5} and {5, (2, 2)}, {4, (2, 1)} is deduced.

(6) Tasks with positive zoning constraint and synchronism constraint.

Both positive zoning constraint and synchronism constraint should be fully satisfied and neither of them can be violated. So we define a new task set named positive-synchronism set. Tasks belonging to this set should be assigned to the corresponding mated-station together.

By the proposed constraint adjustment, the tasks with two of the three additional constraints can be dealt with, and so as the tasks with more than two constraints. For example, for a task with positive zoning constraint and synchronism constraint and positional constraint, we consider positive zoning constraint and synchronism constraint first, then positive zoning constraint and positional constraint, finally synchronism constraint and positional constraint. The positional constraint is left behind because it is allowed to be violated.

4.3.3 Decoding for TALB-MC

In order to obtain a solution satisfying all constraints except for positional constraints, at first we get a candidate task set in which all tasks satisfy the cycle time and precedence constraints. Then, choose the task with the highest priority from the candidate set, and check whether or not the zoning constraints and the synchronism constraints are satisfied for the chosen task. If the answer is true, assign the task(s) to the corresponding station and get a new candidate task set. If not, delete the task and choose another task from the candidate set. If the candidate task set is empty, then open a new mated-station. This process is outlined in Fig. 3.

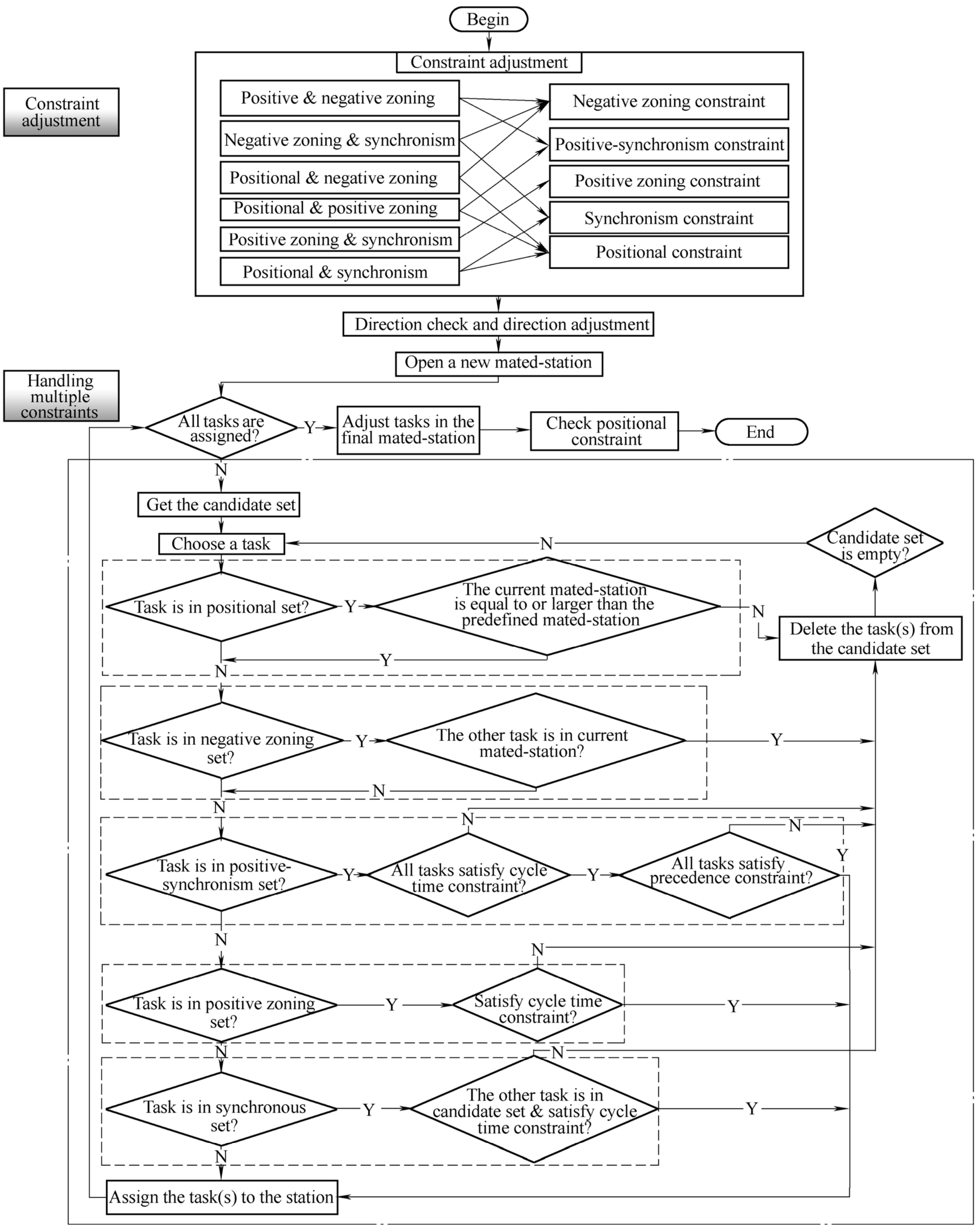
Begin
Constraint adjustment
Constraint adjustment
Positive & negative zoning
Negative zoning & synchronism
Positional & negative zoning
Positional & positive zoning
Positive zoning & synchronism
Positional & synchronism
Negative zoning constraint
Positive-synchronism constraint
Positive zoning constraint
Synchronism constraint
Positional constraint
Direction check and direction adjustment
Open a new mated-station
Handling multiple constraints
All tasks are assigned?
Y
Adjust tasks in the final mated-station
Check positional constraint
End
N
Get the candidate set
Choose a task
N
Candidate set is empty?
Task is in positional set?
Y
The current mated-station is equal to or larger than the predefined mated-station
N
Delete the task(s) from the candidate set
Y
N
Task is in negative zoning set?
Y
The other task is in current mated-station?
Y
N
N
N
N
Task is in positive-synchronism set?
Y
All tasks satisfy cycle time constraint?
Y
All tasks satisfy precedence constraint?
Y
N
N
Task is in positive zoning set?
Y
Satisfy cycle time constraint?
Y
N
N
Task is in synchronous set?
Y
The other task is in candidate set & satisfy cycle time constraint?
Y
N
Assign the task(s) to the station

Fig. 3. Decoding process

Some marks are given as follows.

(1) For tasks with either direction, assign those tasks to the side of the mated-station which can finish the tasks earlier, or select the side randomly when the finish times of both sides are equal.

(2) When we get a solution, the tasks in the last mated-station can be re-assigned to only one side of the last mated-station, under the following conditions: 1) both sides of the last mated-station is used; 2) the tasks assigned to the last mated-station are not involved in positional constraints, the zoning constraints and the synchronism constraints; 3) their operation directions are compatible (i.e., L and E, or R and E); 4) their total operation time is not greater than the cycle time.

(3) In the feasible solution, tasks with the positional constraint must be assigned to predetermined stations. However, when the positional constraints are allowed to be violated, the tasks can be assigned to the predetermined stations, the latter stations or the former mated-station. So in our decoding scheme, the tasks are prevented to be assigned to the former mated-station, and allowed to be assigned to the predetermined stations and the latter stations only, so as to increase the opportunity of finding a solution satisfying the positional constraints to some extent.

(4) For the tasks in the positive-synchronism set, the set of tasks can be assigned under the following conditions: 1) all the tasks satisfy cycle time constraint; 2) all predecessor of the tasks in synchronism set except for tasks in positive zoning set has been assigned.

(5) For the tasks in the positive zoning set, if the total time of the tasks in the set satisfies the cycle time constraint, then the set of tasks can be assigned to the preferred direction. If not, choose another task from the candidate set.

An example of getting a solution for 9-task problem (CT=4) is given as follows. Table 4 displays the generation of a feasible solution based on the corresponding task permutation S (1, 2, 3, 5, 6, 4, 9, 8, 7). And the corresponding task assignments are shown in Fig. 4.

Table 4. An example of generation of a feasible solution of 9-task problem

Mated station	Step	Candidate task set	Choose a task	Constraint check	Description
–	1	–	–	–	Constraint adjustment by step 1
–	2	–	–	–	Direction adjustment by step2
1	3	–	–	–	Open a new mated-station
1	4	{1, 2, 3}	1	Yes	Assign task 1 to (1, 1)
1	5	{2, 3, 4}	2	Yes	Assign task 1 to (1, 2)
1	6	{3, 4, 5}	3	Yes	Assign task 3 to (1, 1)
2	7	–	–	–	Open a new mated-station
2	8	{4, 5, 6}	5	Yes	Assign task 4 to (2, 1)and task 5, 6 to (2, 2)
2	9	{8, 9}	9	No	Assign task 9 to (2, 2)
3	10	–	–	–	Open a new mated-station
3	11	{7, 8}	8	No	Assign task 8 to (3, 1)
3	12	{7}	7	Yes	Assign task 7 to (3, 2)
3	13	–	–	–	Reassign task 8, 7 to (3, 1)

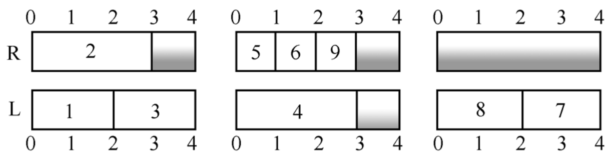

Fig. 4. Task assignments of 9-task in a two-sided assembly line (CT=4)

4.4 Improving solutions by VNS

Variable neighborhood search (VNS) is famous meta-heuristics and in this algorithm different types of neighborhood search structure (NSS) based on different neighborhoods are employed[24]. When a systematic switch from one type of NSS to another happens, there is more opportunity of finding a better solution. The NSS works on the permutation to modify the assignment probability of tasks. In this paper, the neighborhoods are divided into three types, swap operator ($N_1(S)$), multi swap operator ($N_2(S)$) and multi single point operator ($N_3(S)$).

(1) For swap operator:

1) Select task i_1 and task i_2 $(i_1 \neq i_2)$ randomly.

2) Exchange task i_1 and task i_2.

(2) For multi swap operator:

1) Select task i_1 and task i_2 randomly.

2) Exchange task i_1 and task i_2.

3) Repeat above two steps more than twice, which means selecting two other tasks i_3 and i_4 randomly and exchanging them after exchanging task i_1 and task i_2.

Taking the problem in Fig. 2 as an example, the second NSS is depicted in Fig. 5.

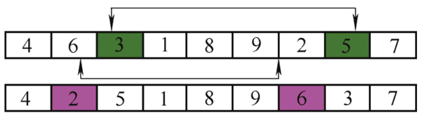

Fig. 5. The second NSS

(3) For multi single point operator:

1) Select task i_1 and task i_2 randomly.

2) Select two sequence positions r_1 and r_2 ($r_1 \neq r_2$) that are different from the positions of task i_1 and task i_2 randomly.

3) Move task i_1 to position r_1 and move task i_2 to position r_2 in the sequence.

4) When it is necessary, rearrange the position of other tasks.

Taking the problem in Fig. 2 as an example, the third NSS is illustrated in Fig. 6.

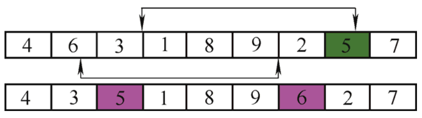

Fig. 6. The third NSS

As for the first NSS ($N_1(S)$), only two different tasks are exchanged. The second NSS ($N_2(S)$) repeats the first NSS more than twice, which of course takes more time. The third NSS($N_3(S)$) needs more time to transfer the original positions of two tasks to the new positions selected randomly, and it can be used only if no improvement is made by the first and the second NSS. In the hybrid TLBO, the first NSS is used at first. Then the second NSS is utilized if we get no improvement about the best solution, finally the third NSS.

4.5 Cost function

The objective of the TALB-MC is minimizing the number of the mated-stations and the number of stations simultaneously within the given cycle time. After the decoding scheme, we may get a solution which violates the positional constraints. Therefore, we take this factor into account and get the following cost function:

$$f = w_{nm}nm + w_{ns}ns + w_{np}np, \tag{25}$$

where nm is the number of mated-stations, ns is the number of stations, and np is the number of violated positional constraints. w_{nm} and w_{ns} are the weighted coefficient of the number of mated-station and the number of stations respectively. w_{np} is the penalty coefficient of the number of the violated positional constraints. Since two stations compose a mated-position, the w_{nm} and w_{ns} are set equal to 2 and 1 respectively. As for w_{np}, we set it 100 so to get a solution which satisfies positional constraints. w_{np} should be large enough, if it is too small, the cost function may have little influence on the solutions.

4.6 Main body of the hybrid TLBO

The hybrid algorithm must attain a balance between the exploration and the exploitation. In this hybrid TLBO, the main role of the TLBO is to explore the searching space, and the general TLBO is modified by introducing the random-keys rule to convert the individual to the task permutation. The principal role of VNS is to exploit the individual obtained by the global TLBO, and three kinds of neighborhood structures are presented in VNS to obtain promising results. The main body of the hybrid TLBO is shown by the following pseudo-code.

Algorithm 1. Hybrid TLBO pseudo-code

```
Begin
  $k \leftarrow 0$; // $k$ is iteration counter
  Population initialization ($P, pop\_size$)
  Fitness evaluation
  While $k < Iter_{Max}$ do // $Iter_{Max}$ is number of the iterations
      $Elite \leftarrow$ select best (elite)
      For $i = 1 \rightarrow pop\_size$ do
          //Teacher Phase
          $T_F = round(1 + rand(0,1))$
          $X_{\text{mean}} \leftarrow$ calculate mean value of $pop\_size$
          $X_{\text{teacher}} \leftarrow$ best solution
          $X_i^{\text{new}} = X_i + r(X_{\text{teacher}} - (T_F X_{\text{mean}}))$
          Evaluate($X_i^{\text{new}}$)
          If $X_i^{\text{new}}$ is better than $X_i$ then
              $X_i \leftarrow X_i^{new}$
          End if // End of Teacher Phase
          //Learner Phase
          $ii \leftarrow random(pop\_size)\{i \neq ii\}$
          If $X_i$ is better than $X_{ii}$ then
              $X_i^{\text{new}} = X_i + r(X_{ii} - X_i)$
          else
              $X_i^{\text{new}} = X_i + r(X_i - X_{ii})$
          End if
          Evaluate($X_i^{\text{new}}$)
          If $X_i^{\text{new}}$ is better than $X_i$ then
              $X_i \leftarrow X_i^{\text{new}}$
          End if // End of Learner Phase
          // Variable neighborhood search
          Diversify $X_i$ and get $X_i^{\text{new}}$ by VNS
          Evaluate ($X_i^{\text{new}}$)
          If $X_i^{\text{new}}$ is better than $X_i$ then
              $X_i \leftarrow X_i^{\text{new}}$
          End if // End of VNS
      End for
      Replace the worst solution with $Elite$
      $k = k + 1$
  End while
  Return best solution
End
```

In this hybrid TLBO algorithm, TLBO is used to find optimal solutions at first. When we can't get better $X_{teacher}$ by TLBO, the VNS will be used to find better solution locally. The first NSS ($N_1(S)$) is used at first and the second NSS is adopted if we get no improvement about the best solution, then the third NSS will be used.

5 Experimental Design and Results

5.1 Experimental design

To test the performance of the proposed algorithm, the hybrid TLBO was coded in C++ programming language with the software of Microsoft Visual C++ 6.0. All tests are conducted on an Intel(R) Core2(TM) CPU 2.33 GHZ, 3.036 GB RMA personal computer using Microsoft Windows XP. A set of test problems with different cycle times are solved: four small-size problems, P9, P12, P16 and P24; three large-size problems P65, P148 and P205. P9, P12, and P24 are taken from KIM, et al[3], P16, P65 and P205 are taken from LEE et al[2], and P148 is taken from BARTHOLDI[1], and the operation times of task 79 and task 108 are modified by LEE, et al[2]. All the test cases are well-known benchmark problems for TALB problem and notation P148 means the TALB problem with 148 tasks. Additional constraints for all the instances are taken from BIAO, et al[22]. Note that tasks

{131(L), 137(L)} for P148 are changed to {131(L), 132(E)} with confirmation from the author since tasks with synchronism constraints must be operated at the opposite side of the same mated-station. The parameters of the proposed algorithm are determined by preliminary experiments. Since the sizes of the problems are different from each other, the numbers of iteration are 100 for small-sized problems and 300 for large-sized problems.

5.2 Experimental results comparisons without multiple constraints

The results comparison of two-sided assembly lines is shown in Table 5. The evaluation criteria including the number of stations (*ns*), the number of mated-stations (*nm*) and the CPU time (*s*) are reported. The best, average and maximum numbers of stations among 20 times are reported by *Min*, *Avg* and *Max*, respectively.

Table 5. Computational results without multiple constraints

Problem	Cycle time	LB	ACO	TS	2-ANTBAL	BA	TLBO				Hybrid TLBO				CPU times
							ns			*nm*	*ns*			*nm*	*s*
							Min	*Avg*	*Max*		*Min*	*Avg*	*Max*		
P9	3	6	6	6	–	6	6	6	6	3	**6**	6	6	3	0.13
	4	5	5	5	–	5	5	5	5	3	**5**	5	5	3	0.11
	5	4	4	4	–	4	4	4	4	2	**4**	4	4	2	0.11
	6	3	3	3	–	3	3	3	3	2	**3**	3	3	2	0.11
P12	4	7	–	–	–	7	7	7	7	4	**7**	7	7	4	0.14
	5	5	6	6	–	6	6	6	6	3	**6**	6	6	3	0.13
	6	5	5	5	–	5	5	5	5	3	**5**	5	5	3	0.13
	7	4	4	4	–	4	4	4	4	2	**4**	4	4	2	0.13
	8	4	–	4	–	4	4	4	4	2	**4**	4	4	2	0.13
P16	15	6	–	–	–	6	6	6	6	4	**6**	6	6	4	0.17
	16	6	–	6	–	6	6	6	6	3	**6**	6	6	3	0.17
	18	5	–	–	–	6	6	6	6	3	**6**	6	6	3	0.17
	19	5	–	5	–	5	5	5	5	3	**5**	5	5	3	0.17
	20	5	–	–	–	5	5	5	5	3	**5**	5	5	3	0.17
	21	4	–	5	–	5	5	5	5	3	**5**	5	5	3	0.17
	22	4	–	4	–	4	4	4	4	2	**4**	4	4	2	0.17
P24	18	8	–	8	–	8	8	8	8	4	**8**	8	8	4	0.28
	20	7	8	8	–	8	8	8	8	4	**8**	8	8	4	0.28
	24	6	–	6	–	6	6	6.5	7	3	**6**	6.3	7	3	0.26
	25	6	6	6	–	6	6	6	6	3	**6**	6	6	3	0.27
	30	5	5	5	–	5	5	5	5	3	**5**	5	5	3	0.26
	35	4	5	4	–	4	4	4	4	2	**4**	4	4	2	0.25
	40	4	4	4	–	4	4	4	4	2	**4**	4	4	2	0.26
P65	326	16	17	17	17	17	17	17	17	9	**17**	17	17	9	5.85
	381	14	15	15	14	14	14	14.9	15	7	**14**	14.9	15	7	5.56
	435	12	13	13	13	12	13	13	13	7	13	13	13	7	5.57
	490	11	12	11	12	11	11	11	11	6	**11**	11	11	6	5.62
	544	10	10	10	10	10	10	10	10	5	**10**	10	10	5	5.61
P148	204	26	26	26	26	26	26	26.8	27	13	**26**	26.8	27	13	10.01
	255	21	21	21	21	21	21	21.1	22	11	**21**	21.4	22	11	9.84
	306	17	18	18	18	17	18	18	18	9	18	18	18	9	9.76
	357	15	15	15	15	15	15	15	15	8	**15**	15.2	16	8	9.04
	408	13	14	13	14	13	13	13.8	14	7	**13**	13.5	14	7	9.23
	459	12	12	12	12	12	12	12	12	16	**12**	12	12	6	9.85
	510	11	11	11	11	11	11	11	11	6	**11**	11	11	6	9.32
P205	1133	21	24	24	22	22	22	23.1	24	11	**22**	22.9	23	11	16.11
	1322	18	22	21	20	20	20	20	20	10	**19**	19.8	20	10	15.42
	1510	16	18	18	17	17	17	17.7	18	9	**17**	17.3	18	9	16.24
	1699	14	18	17	15	16	15	15.6	16	8	**15**	15.6	16	8	16.03
	1888	13	15	16	13	14	14	14	14	7	14	14	14	7	16.61
	2077	12	14	14	12	12	13	13	13	6	**12**	12.9	13	6	15.29
	2266	11	12	13	12	12	12	12	12	6	**12**	12	12	6	15.83
	2454	10	12	12	10	11	11	11.3	12	6	11	11.1	12	6	14.67
	2643	9	11	11	10	10	10	10	10	5	**10**	10	10	5	15.46
	2832	9	10	10	10	10	10	10	10	5	**9**	9.9	10	5	16.15

The *nm* in Table 5 is the mated-station for the best number of stations. Table 5 summarizes the results obtained by an ant colony-based heuristic (ACO) algorithm by BAYKASOGLU, et al[4], Tabu Search Algorithm (TS) developed by ÖZCAN, et al[13], an ant colony optimization algorithm (2-ANTBAL) by SIMARIA, et al[5], Bee colony intelligence (BA) by ÖZBAKIR, et al[19]. LB is obtained using the lower bound (LB) introduced by WU, et al[11] on the number of stations for TALB-I. The lower bounds are calculated by the following equations:

$$S_L = \left\lceil \sum_{i\in AL} t_i / CT \right\rceil, \tag{26}$$

$$S_R = \left\lceil \sum_{i\in AR} t_i / CT \right\rceil, \tag{27}$$

$$S_E = \left\lceil \max\left(\left(\sum_{i\in AE} t_i - (LB_{\text{left}} + LB_{\text{right}})CT - \sum_{i\notin AE} ti\right), 0\right) / CT \right\rceil, \tag{28}$$

$$LB = S_L + S_R + S_E, \tag{29}$$

where S_L, S_R and S_E are the minimum number of stations for the tasks with the left, right and either direction, respectively.

In Table 5, among 45 instances the proposed hybrid TLBO outperforms in 13 cases over ACO and 11 over TS. Especially for the largest problem, P205, which comes from real application, 9 or 10 out of 10 cases are outperformed by the proposed hybrid algorithm over ACO or TS, respectively. In addition, the proposed hybrid TLBO outperforms in 4 cases over 2-ANTBAL and 3 over BA. As for P65 (CT=435) and P148 (CT=306), the hybrid TLBO get worse results compared with BA, but we still get the same results compared with 2-ANTBAL. As for P205 (CT=1888, 2454), we get worse results compared with 2-ANTBAL, but the results of hybrid TLBO are still same to the results from BA. And the hybrid TLBO outperforms TLBO and the hybrid TLBO has more chance to find optimal or near optimal solutions. Actually, the local optimal solutions within each population are improved by VNS when no better global solution can be obtained by TLBO. Thus the proposed hybrid TLBO is able to find optimal and near optimal solution within the limited number of iterations. As for the computational times, all of them are less than 1 min including the largest case.

According to these figures in Table 5, the proposed algorithm performed well for the large-size problems with different cycle times and the deviation is acceptable. As for the small-size problems, the TLBO and the hybrid TLBO algorithm get all the best solutions among the algorithms. The computational studies demonstrate that the TLBO and the hybrid TLBO algorithm outperform ACO and TS, especially for the large-sized problems.

5.3 Experimental results comparisons with multiple constraints

To show the influence of multiple constraints, the Gantt chart of an optimal schedule for P148 (CT=459) is shown in Fig. 7. It is obvious that the total time of all tasks on each station is less than the cycle time and station S12 has been allocated with the least workload. Due to synchronism constraints {131, 132}, tasks 132 have to wait on S10 until task 131 can start. On the other hand, task 54, 123, 109 cannot be operated until task 55, 113, 108 has been completed because of sequence dependence.

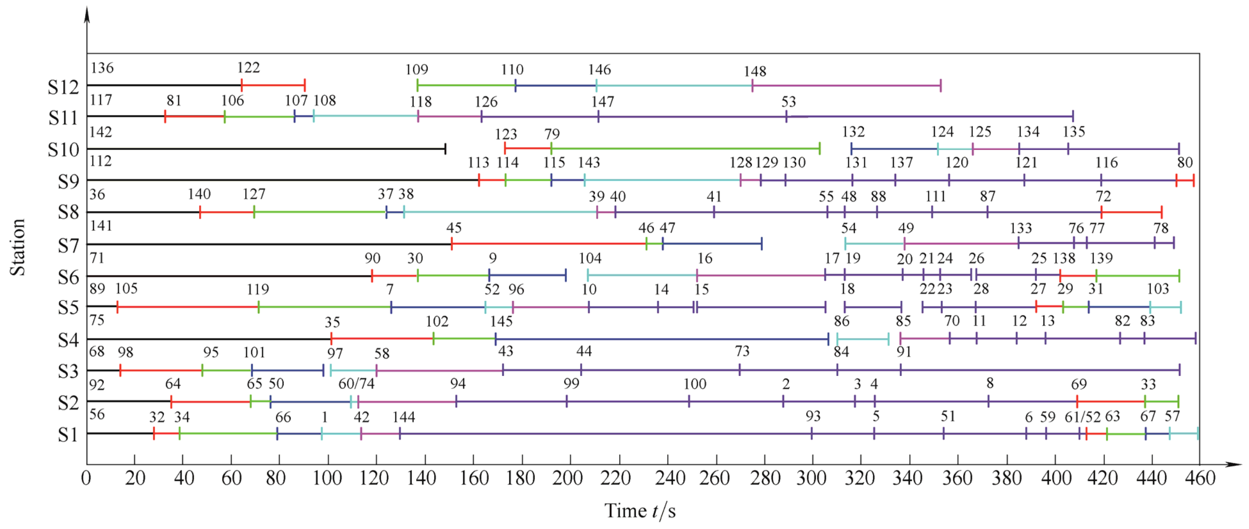

Fig. 7. Gantt chart of an optimal schedule for P148 (CT=459)

The performance of the hybrid TLBO is compared with that of IP and the late acceptance hill-climbing algorithm (LAHC) by BIAO, et al[22]. The number of stations (ns), the number of mated-stations (nm) and the CPU time (s) are reported in Table 6. The best, average and maximum solutions of ns among 20 times are reported by Min, Avg and Max, respectively. The nm in Table 6 is the mated-station for the best solution.

From Table 6, for small-size problems, the proposed algorithm can get almost all the best solutions except for P16

(*CT*=7). As for P16 (*CT*=7), we get the optimal number of the mated-station. In the decoding scheme, E-type tasks with either direction should be assigned to the side of the mated-station which can finish the tasks earlier. Therefore, it is difficult to find a solution with 6 stations and 4 mated-stations. Among the 15 cases for large-size problems, 9 cases are outperformed by the proposed algorithm over LAHC. In addition, the computational times are shorter than 1 min, even for the largest case. Hence, the proposed hybrid TLBO algorithm outperforms LAHC in most cases. In summary, the hybrid TLBO demonstrates that it is efficient and effective for solving the TALB-MC problems.

Table 6. Computational results with multiple constraints

Problem	Cycle time	LB	IP *nm*[*ns*]	LAHC *nm*[*ns*]	Hybrid TLBO				CPU *s*
					ns			*nm*	
					Min	*Avg*	*Max*		
P9	3	6	4[7]	4[7]	7	7	7	4	0.12
	4	5	3[5]	3[5]	5	5	5	3	0.15
	5	4	2[4]	2[4]	4	4	4	2	0.14
	6	3	2[3]	2[3]	3	3	3	2	0.14
	7	3	2[3]	2[3]	3	3	3	2	0.14
P12	5	5	3[6]	3[6]	6	6	6	3	0.14
	6	5	3[5]	3[5]	5	5	5	3	0.14
	7	4	3[5]	3[5]	5	5	5	3	0.14
	8	4	2[4]	2[4]	4	4	4	2	0.14
	9	3	2[4]	2[4]	4	4	4	2	0.14
P16	15	6	4[7]	4[7]	7	7	7	4	0.14
	16	6	4[6]	4[6]	7	7	7	4	0.14
	18	5	3[6]	3[6]	6	6	6	3	0.15
	20	5	3[5]	3[5]	5	5	5	3	0.16
	21	4	3[5]	3[5]	5	5	5	3	0.16
P24	20	7	–	4[8]	8	8	8	4	0.43
	25	6	–	3[6]	6	6	6	3	0.42
	30	5	–	3[5]	5	5	5	3	0.41
	35	4	–	3[5]	5	5	5	3	0.42
	40	4	–	2[4]	4	4	4	2	0.42
P65	326	16	–	10[20]	**17**	17.4	18	9	4.54
	381	14	–	8[16]	**15**	15	15	8	4.91
	435	12	–	7[14]	**13**	13	13	7	5.37
	490	11	–	6[12]	12	12	12	6	5.96
	544	10	–	6[11]	11	11	11	6	5.76
P148	255	21	–	13[25]	**22**	22.8	23	11	9.74
	306	17	–	11[21]	**19**	19	19	9	9.68
	357	15	–	9[17]	**16**	16	16	8	9.63
	459	12	–	7[14]	**12**	12.6	13	6	9.58
	510	11	–	7[13]	**11**	11	11	6	9.68
P205	1888	13	–	8[15]	**14**	14.9	15	7	16.04
	2266	11	–	7[13]	13	13	13	7	16.01
	2454	10	–	6[12]	12	12	12	6	16.86
	2643	9	–	6[11]	11	11.2	12	6	16.39
	2832	9	–	6[11]	11	11	11	6	16.25

In fact, the TLBO is used for global search and the VNS is used to search better solutions locally, which makes this algorithm achieve the right balance between intensification and diversification. By the proposed direction adjustment, the computational effort has been reduced. For tasks with the positional constraints, the tasks should be in the predetermined mated-station or the later mated-station, and the former mated-station is refused to reduce search space. By the way, there is larger possibility to find a solution which satisfies all the constraints.

6 Conclusions

(1) The multiple constraints, including the positional constraints, zoning constraints and synchronism constraints are considered, which are usually found in real application.

(2) A decoding scheme is proposed to handle tasks with several additional constraints and we can get a solution which satisfies zoning constraints and the synchronism constraints. As for the positional constraints, the search space is reduced by assigning the tasks with positional constraints to the predetermined mated-station and the later mated-station, and we have more chance to find a solution which satisfies all the additional constraints.

(3) A hybrid algorithm is proposed to solve the two-sided assembly problems with multiple constraints. The TLBO algorithm is utilized to search for global optimum, and VNS is employed to search for better local

optimal solutions by intensifying task permutation. By hybridizing TLBO and VNS, the performance is improved and it has more chances to find optimal or near optimal solutions.

References

[1] BARTHOLDI J J. Balancing two-sided assembly lines: a case study[J]. *International Journal of Production Research*, 1993, 31: 2447–2461.

[2] LEE T O, KIM Y, KIM Y K. Two-sided assembly line balancing to maximize work relatedness and slackness[J]. *Comput. Ind. En.*, 2001, 40(3): 273–292.

[3] KIM Y K, KIM Y, KIM Y J. Two-sided assembly line balancing: a genetic algorithm approach[J]. *Prod Plan Control*, 2000, 11(1): 44–53.

[4] BAYKASOGLU A, DERELI T. Two-sided assembly line balancing using an ant-colony-based heuristic[J]. *Int. J. Adv. Manuf. Technol.*, 2008, 36: 582–588.

[5] SIMARIA A S, VILARINHO P M. 2-ANTBAL: An ant colony optimisation algorithm for balancing two-sided assembly lines[J]. *Computers & Industrial Engineering*, 2009, 56(2): 489–506.

[6] RAO R V, SAVSANI V J, VAKHARIA D P. Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems[J]. *Computer-Aided Design*, 2011, 43(3): 303–315.

[7] PAWAR P J, RAO R V. Parameter optimization of machining processes using teaching–learning-based optimization algorithm[J]. *Int. J. Adv. Manuf. Technol.*, 2013, 67: 995–1006.

[8] RAO R V, SAVSANI V J, VAKHARIA D P. Teaching-learning-based optimization: an optimization method for continuous non-linear large scale problems[J]. *Information Sciences*, 2012, 183(1): 1–15.

[9] MLADENOVIC N, HANSEN P. Variable neighborhood search[J]. *Computers & Operations Research*, 1997, 24: 1097–1100.

[10] HU X, ERFEI W, YE J. A station-oriented enumerative algorithm for two-sided assembly line balancing[J]. *European Journal of Operational Research*, 2008, 186: 435–440.

[11] WU E F, JIN Y, BAO J S, HU X F. A branch-and-bound algorithm for two-sided assembly line balancing[J]. *International Journal of Advanced Manufacturing Technology*, 2008, 39: 1009–1015.

[12] KIM Y K, SONG W S, KIM J H. A mathematical model and a genetic algorithm for two-sided assembly line balancing[J]. *Computers & Operations Research*, 2009, 36(3): 853–865.

[13] ÖZCAN U, TOKLU B. A tabu search algorithm for two-sided assembly line balancing[J]. *International Journal of Advanced Manufacturing Technology*, 2009, 43: 822–829.

[14] ÖZCAN U, TOKLU B. Multiple-criteria decision-making in two-sided assembly line balancing: A goal programming and a fuzzy goal programming models[J]. *Computers & Operations Research*, 2009, 36(6): 1955–1965.

[15] ÖCAN U, TOKLU B. Balancing of mixed-model two-sided assembly lines[J]. *Computers & Industrial Engineering*, 2009, 57(1): 217–227.

[16] HU X F, WU E F, BAO J S, et al. A branch-and-bound algorithm to minimize the line length of a two-sided assembly line[J]. *European Journal of Operational Research*, 2010, 206(3): 703–707.

[17] ÖZCAN U. Balancing stochastic two-sided assembly lines: A chance-constrained, piecewise-linear, mixed integer program and a simulated annealing algorithm[J]. *European Journal of Operational Research*, 2010, 205(1): 81–97.

[18] ÖZBAKIR L, TAPKAN P. Balancing fuzzy multi-objective two-sided assembly lines via bees algorithm[J]. *Journal of Intelligent & Fuzzy Systems*, 2010, 21(5): 317–329.

[19] ÖZBAKIR L, TAPKAN P. Bee colony intelligence in zone constrained two-sided assembly line balancing problem[J]. *Expert Systems with Applications*, 2011, 38(9): 11 947–11 957.

[20] TAPKAN P, OZBAKIR L, BAYKASOGLU A. Bees algorithm for constrained fuzzy multi-objective two-sided assembly line balancing problem[J]. *Optimization Letters*, 2012, 6(6): 1039–1049.

[21] CHUTIMA P, CHIMKLAI P. Multi-objective two-sided mixed-model assembly line balancing using particle swarm optimisation with negative knowledge[J]. *Computers & Industrial Engineering* , 2012,62(1): 39–55.

[22] BIAO Y, CHAOYONG Z, XINYU S. A late acceptance hill-climbing algorithm for balancing two-sided assembly lines with multiple constraints[J]. *J. Intell. Manuf.*, 2015, 26(1): 159–168.

[23] BEAN J. Genetics and random keys for sequencing and optimization[J]. *ORSA Journal on Computing*, 1994, 6(2): 154–160.

[24] HANSEN P, MLADENOVIC N. Variable neighborhood search: Principles and applications[J]. *European Journal of Operational Research*, 2001, 130: 449–467.

Biographical notes

TANG Qiuhua, born in 1970, is currently a professor at *Wuhan University of Science and Technology, China*. She received her PhD degree from *Wuhan University of Science and Technology, China*, in 2005. Her research interests include production planning and scheduling and industrial engineering.
E-mail: tangqiuhua@wust.edu.cn

LI Zixiang, born in 1990, is currently a postgraduate at *Wuhan University of Science and Technology, China*. He received his bachelor degree from *Wuhan University of Science and Technology, China*, in 2013.

ZHANG Liping, is currently a lecturer at *Wuhan University of Science and Technology, China*. She received her PhD degree from *Huazhong University of Science and Technology, China*, in 2013. Her research interest include dynamic scheduling, job shop scheduling and intelligent algorithm.

FLOUDAS C A, is currently a professor at *Texas A&M University, USA*. His research interests include product and process systems engineering, and bioinformatics and computational genomics.

CAO Xiaojun, born in 1974, is currently a senior engineer at *Technique Center of Dongfeng Peugeot Citroen Automobile Company, China.* He received his bachelor degree from *Huazhong University of Science and Technology, China*, in 1996.