International Journal of Mechanical, Industrial and Aerospace Sciences
ISSN: 2517-9950
Vol:8, No:1, 2014

# Applying Branch-and-Bound and Petri Net Methods in Solving the Two-Sided Assembly Line Balancing Problem

Nai-Chieh Wei, I-Ming Chao, Chin-Jung Liuand, Hong Long Chen

***Abstract*—**This paper combines the branch-and-bound method and the petri net to solve the two-sided assembly line balancing problem, thus facilitating effective branching and pruning of tasks. By integrating features of the petri net, such as reachability graph and incidence matrix, the propose method can support the branch-and-bound to effectively reduce poor branches with systematic graphs. Test results suggest that using petri net in the branching process can effectively guide the system trigger process, and thus, lead to consistent results.

***Keywords*—**Branch-and-Bound Method, Petri Net, Two-Sided Assembly Line Balancing Problem.

## I. Introduction

THE two-sided assembly line has two parallel production lines, on the left and right sides. In the production line for a single product, tasks can be simultaneously assigned to parallel stations, and some specific tasks can be defined for assignment to a specific side. For example, if a task is defined as a left side (L) or right side (R) task, it can be assigned to the station of the left side or right side of the production line accordingly. Tasks defined as either side (E) can be assigned to either the left side or right side of the production line. Moreover, tasks can be appropriately assigned to stations on both sides for synchronized assembly, thus reducing idle time without conflicting with the requirement. The stations of the left and right sides of the production line are known as mated-stations (positions), and are of the same in the cycle time, meaning one side has a companion relationship with the station on the other side. In Fig. 1, the number in brackets represent task time, the English alphabets represent the task assignment direction (L: left side; R: right side; E: either side), the numbers in circles represent task number, the arc is the process assembly precedence, and the total task time is 82 units [1]-[2].

N. C. Wei, I. M. Chao, and C. J. Liu are with the Department of Industrial Management, I-Shou University, Kaohsiung City 84001, Taiwan, R.O.C. (phone: +886-657-7711; fax: +886-657-8536; e-mail: ncwei@ isu.edu.tw).
H. L. Chen is with the Department of Business and Management, National University of Tainan, Tainan 700, Taiwan, R.O.C.

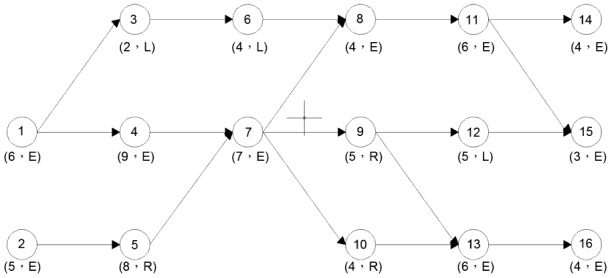


Fig. 1 A precedence diagram

Assuming cycle time is 15 units, the optimal task assignment solution is of 6 stations for the single-sided production line, while the two-sided assembly line requires only 4 positions, as shown in Fig. 2 [3]. In comparison with the single-sided production line, the advantages of the two-sided assembly line include shorten the production line length, reduce material handling costs, and reduce the movement times of workers; thereby increasing productivity. Moreover, as the mated-stations of both sides share tools and equipment, which could reduce purchase costs, it is often applied to large-scale product manufacturing processes such as vehicles [4]-[7].

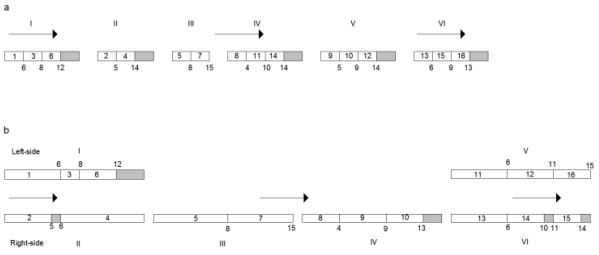


Fig. 2 Configuration difference between a single-sided and a two-sided assembly line

Regarding the algorithms proposed by [3], the branch-and-bound method provides task assignments during the computational process in a graphical manner. The only disadvantage of this method is that it cannot effectively cut the branches of the node, resulting in cumbersome computational processes. The solution efficiency of the branch-and-bound method is determined by branching efficiency and pruning ability [8]. A well-designed branching strategy can effectively reduce nodes branches, and improve solution efficiency. According to [9], the search is far more complex than the

single-sided production line in branching E-type tasks using the branch-and-bound method for the two-sided assembly line.

To quickly reduce excessive branching, when applying the branch-and-bound method to solve the problem of the two-sided assembly line, the task selection and triggering steps should be able to list the feasible routes of task assignments, and provide the constraints and feedback tracing. Thus, it is more applicable to small-scale problems. However, as it may increase the assignment complexity when applied to large-scale problems, there is a need to combine with other methods to speed up the process. In this regard, the incidence matrix and reachability graph of petri net can meet the need. The reachability graph can clearly demonstrate the entire event status, the relationship between tasks, and proceeding direction; while the incidence matrix can represent the relationship degree of tasks and events, and induce the entire process by numbers. Hence, when properly used, they can effectively help the branch-and-bound method to improve the efficiency of the branching process.

## II. Review of Petri Net

Petri net includes token, place, transition, and arc. The flow of tokens in the system represents the dynamic behavioral pattern of the system. Triggering the temporary state can result in the flow of tokens and change the markings of petri net. The basic petri net is a tool combining mathematics and graphs, and is used in system design. It is a diagram of circle, rectangle, and arc. In the assembly line, transitory triggering represents an action. The combination of place and transition represents the precedence of certain situations and relationships. The token of place represents the action of the moment. Hence, the production process information or action can be tracked by tokens of the system. The two standard features of petri net are: reachability graph and incidence matrix.

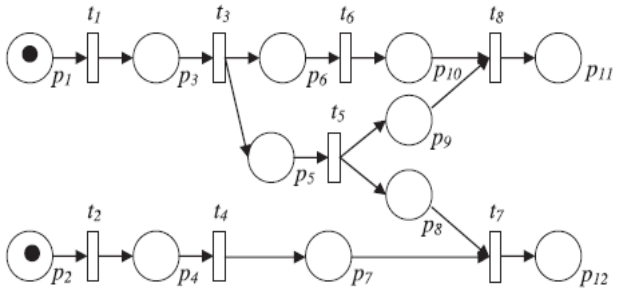


Fig. 3 An example of the reachability graph

$$\mathbf{A}=\begin{array}{c} \begin{matrix} p_1 & p_2 & p_3 & p_4 & p_5 & p_6 & p_7 & p_8 & p_9 & p_{10} & p_{11} & p_{12} \end{matrix} \\ \begin{bmatrix} 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 & 0 \end{bmatrix} \end{array} \begin{matrix} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \\ t_6 \\ t_7 \\ t_8 \end{matrix}$$

Fig. 4 An example of the incidence matrix

The relationships of various tasks with place are as shown in Fig. 3. By converting the relationships into a mathematical matrix, the starting point is M0=[1 1 0 0 0 0 0 0 0 0 0 0] and ending point is ML=[0 0 0 0 0 0 0 0 0 0 1 1]. According to the direction of the process, the task of t1 is in the direction of an arc in relation to P1 and P3. The incoming place has a positive relationship in terms of task, while the outgoing place is the opposite (1: positive relationship; -1: negative relationship). The matrices can be listed according to the above rules. Then, by token triggering and matrix computation, for example, if t1 is triggered, the matrix results of M0-t1=[0 1 1 0 0 0 0 0 0 0 0 0], which represent that tokens have fallen at P2 and P3. Each step of the assembly line can be computed according to the obtained results. In other words, the movement and assignment of tokens can be marked by small points in various places to display the current situation of the system, as shown in Fig. 4 [10].

## III. The Proposed Method

### *A. The Solution Procedure*

Fig. 5 illustrates the basic process of combining the branch-and-bound method and petri net for the mathematical model proposed by [11]. The steps are summarized as follows:

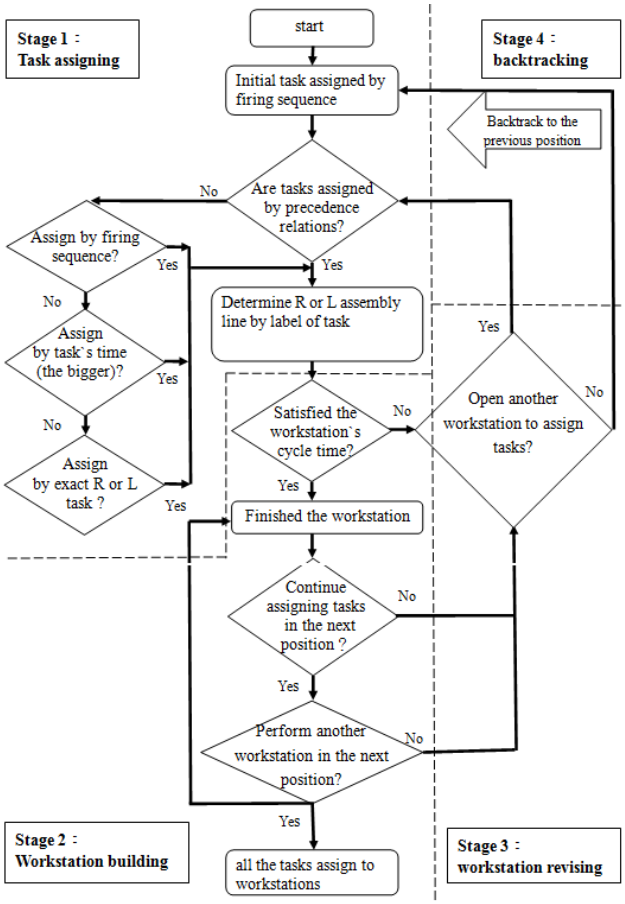


Fig. 5 The solution flowchart

Step 1: Task assigning

1-1 the initial task is determined by the sequence-dependent ending time;

1-2 the branch tasks triggered by the initial task are assigned according to the degree of task relationships;

1-3 the subsequent assignment is based on the degree of task relationships;

1-4 the tasks of the same level of relationship are assigned according to the sequence-dependent ending time;

1-5 the tasks of the same ending time are assigned according to the time length;

1-6 the tasks of the same time length are assigned according to the operational direction of left or right side;

1-7 after assigning tasks to stations according to the above steps, the left and right side stations can be determined by its task type.

Step 2: Workstation building

2-1 the initial tasks of the subsequent stations are assigned according to the ending time of tasks if the left side station task assignment is completed, it is changed to the right side station;

2-2 after satisfying the cycle time of the right side station, the assignment of task for the right side station of the next position can be started;

2-3 after assigning the right side station, the left side station is assigned alternately, and task assignment can be conducted in the same manner.

Step 3: Workstation revising

3-1 if the left side station has no appropriate branch task for triggering, it may set the right side station for the branch task;

3-2 assign the triggered branch tasks to the right side station until the station has no appropriate task assignments;

3-3 due to the branch task triggered by the right side station, the previously stalled left side station may be actuated to assign the task in an alternate manner;

3-4 after completing the left and right side stations of the current position, the task assignment of the stations of the next position can be started.

Step 4: Backtracking

4-1 when the left and right side stations have no appropriate tasks to assign, or the idle time is too long (greater than the backtracking value), it must backtrack to the previous task or the ending point of the previous station;

4-2 perform task assignment according to task assignment rules in order to properly assign all tasks to stations, use petri net for re-branching, and proceed until the station assignment is completed.

*B. Task Assignment Rules*

Rule 1 : Task assignment is based on the degree of relationship, that is the immediate relationship as a priority, followed by the sharing relationship, common relationship, and no relationship.

1) If task a produces branch tasks b and c after triggering, then b and c are the immediate relationships to the upper layer.

2) If tasks b and c can produce branch task d, then d is the sharing relationship of tasks b and c of the upper layer.

3) If branch task a can produce branch task f after triggering, f and a belong to neither relationship; however, they are called a common relationship as they are triggered on the same side.

4) If task g on the right side station is triggered, tokens will fall on tasks c and h for assignment; in this case, task c is called a no relationship to g.

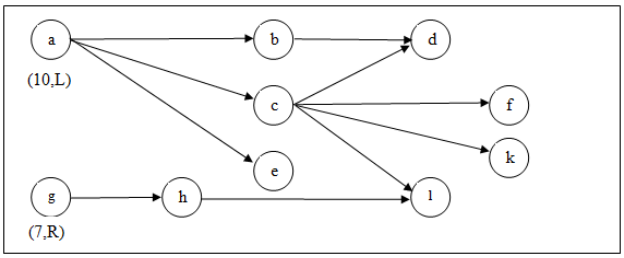


Fig. 6 An example of a precedence diagram

Rule 2 : If a branch task produces a number of tasks of the same degree of relationship, the assignment should be based on the ending time sequence, followed by tasks of longer time length, and tasks of left or right directions.

1) If task c produces branch tasks d, f, k, and l after triggering, tasks f and k are of an immediate relationship, while tasks d and l are of the sharing relationship.
2) Tasks f and k are assigned in priority with the ending time sequence, where time length is the basis of assignment.
3) If the assignments of tasks f and k fail, tasks d and l of the common relationship are assigned according to the ending time sequence and time length.

Rule 3 : Assigning tasks of no relationship according to the latest starting time of tasks triggered in the station. If it is greater than the latest starting time of the station, the gap between the two time periods is idle time. Assignments cannot be made when the idle time is greater than the backtracking value.

1) If the task computation of the left side station amounts to a total of 10 time units, the latest starting time of branch tasks b, c, and e triggered by task a is 10.
2) If the computation of the subsequent right side stations to task g amounts to a total of 7 time units, and task g must assign task c as no relationship, then the right side station can produce three units of idle time (10-7).

Rule 4 : After assigning the station according to the task precedence, it can possibly form the independent operations of single-sided stations (with no mated-station). In the task assignment of the following position, tasks should be assigned to the left side and right side station for alternate manner.

### C. An Illustrative Example

Taking an example as shown in Fig. 1, the firing sequence is obtained according to the ending time of the tasks, 2, 1, 3, 6, 5, 4, 7, 8, 10, 9, 11, 12, 13, 15, 14, 16 (see Fig. 7), and the two features of petri net (as shown in Figs. 8 and 9). If the cycle time is 15, the computation is as follows:

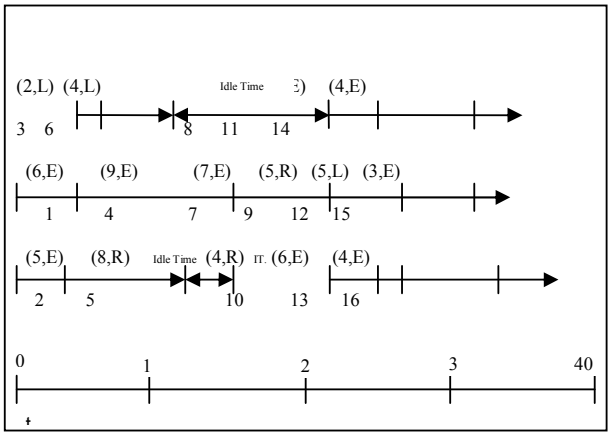


Fig. 7 A diagram of task time sequence

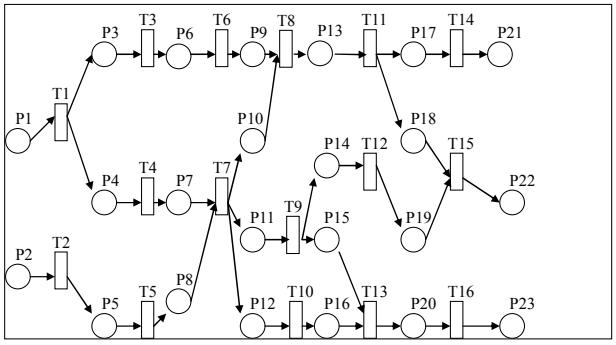


Fig. 8 Reachability graph of the illustrated example

| P / T | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | | -1 | -1 | | | | | | | | | | | | | | | | | | | |
| 2 | | 1 | | | -1 | | | | | | | | | | | | | | | | | | |
| 3 | | | 1 | | | -1 | | | | | | | | | | | | | | | | | |
| 4 | | | | 1 | | | -1 | | | | | | | | | | | | | | | | |
| 5 | | | | | 1 | | | -1 | | | | | | | | | | | | | | | |
| 6 | | | | | | 1 | | | -1 | | | | | | | | | | | | | | |
| 7 | | | | | | | 1 | 1 | | -1 | -1 | -1 | | | | | | | | | | | |
| 8 | | | | | | | | | 1 | 1 | | | -1 | | | | | | | | | | |
| 9 | | | | | | | | | | | 1 | | | -1 | -1 | | | | | | | | |
| 10 | | | | | | | | | | | | 1 | | | | -1 | | | | | | | |
| 11 | | | | | | | | | | | | | 1 | | | | -1 | -1 | | | | | |
| 12 | | | | | | | | | | | | | | 1 | | | | | -1 | | | | |
| 13 | | | | | | | | | | | | | | | 1 | 1 | | | | -1 | | | |
| 14 | | | | | | | | | | | | | | | | | 1 | | | | -1 | | |
| 15 | | | | | | | | | | | | | | | | | | 1 | 1 | | | -1 | |
| 16 | | | | | | | | | | | | | | | | | | | | 1 | | | -1 |
| M0 | 1 | 1 | | | | | | | | | | | | | | | | | | | | | |
| ML | | | | | | | | | | | | | | | | | | | | | 1 | 1 | 1 |

Fig. 9 Incidence Matrix of the illustrated example

As shown in Fig. 9, the assembly line conducts a series of task assignment from the left side to the right side. Therefore, for the relative relationship of T1 and P, P1 enters from the left side, while P3 and P4 come from the right side; hence, P1 is a positive number, and P3 and P4 are negative. The relationship matrix of task (T) and place (P) can be completed in the same manner. In addition, starting point M0(P1, P2) and ending point ML(P21, P22, P23) are both set as positive: M0=[110000000000000000000000] and ML= [000000000000000000000111].

This study develops a backtracking value to determine the success or failure of the station assignment. If the idle time is greater than the backtracking value, backtracking operations should be triggered to reassign the appropriate tasks, and to avoid impacts on the overall balancing efficiency. The backtracking value of the example is 3, and the computation is as follows:

A: the task average time
B: total task time/cycle time= the number of ideal stations
C: B*cycle time=total time of the completion
D: (C-total task time)/B=average station idle time
E: (A+D)/2=backtracking value (rounded for integer)

1. Computational Steps - Task Failure, Execution of Backtracking

In the branch-and-bound method, ROOT is the starting point and symbol (1 T2 1R 5 10) representing the assignment no., task no., left or right station of the current position no., task time, and remaining cycle time to track the task assignment. Moreover, the underlined no. (n) represents the deletion of task assignment according to the firing sequence (as shown in Fig. 10).

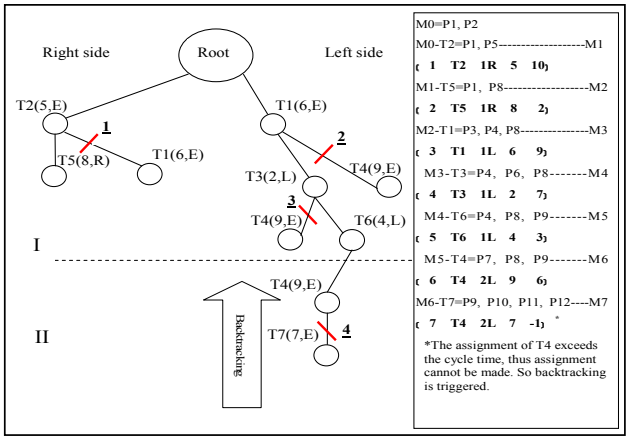


Fig. 10 Computational steps- task assignment failure

After triggering assignment no. 1and T2, the M0 in the matrix can be used to obtain M1 by subtracting T2=[0100-1000000000000000000], and matrix representation is M1=[1000100000000000000]. After triggering token falls at P1 and P5 and branch tasks T1 and T5 become executable (see Figs. 8-10), assignment no. 2 should be carried out. T5 is the "immediate task" of T2, while T1 is a task of "no relation"; therefore, T5 is assigned first. Regarding assignment no. 3, T1 task is assigned according to the task ending time sequence and assignment no. 4. T3 and T4 are the "immediate tasks" of T1, thus, T3 is assigned according to the sequence of ending time. For assignment no. 7, according to the branching results of triggering the T4 task, only the T7 task can be assigned, and the time length of the T7 task is 7 units, which is beyond the remaining cycle time of the station. Thus, the station cannot be constructed. If T4 is assigned, as the idle time is greater than the backtracking value (6>3), it may affect the balance efficiency. The station thus cannot be constructed and it must be backtracked to the previous stage for reassignment, as shown in Fig. 10.

After the assignment failure of T7, it conducts backtracking operation for reassignment. In the process of backtracking, assignments no. 6 and 5 have no other branch tasks for assignment, thus, backtracking is continued. Regarding another branch task of assignment no. 4, if it is beyond the cycle time after upward backtracking, the other branch task of assignment no. 3 (T4) can satisfy the cycle time and complete the current station assignment.

2. Computational Steps - Task Completion

The left and right side stations of the first position can be found by backtracking (as shown in Fig. 11) before carrying out the left side station task assignment of the second position. In assignment no. 5, T3 and T7 are branch tasks, where T3 is triggered according to the ending time sequence. For assignment no. 6, T9 and T10 tasks are triggered by the T7 task, and as it is a right side task (R), hence, it is not assigned. According to Rule 3, when assigning the right side station, T8, T9, and T10 are tasks triggered by left side tasks T7 and T6. If the assigned idle time of the stations is the latest ending time of T7 (7) and T6 (13), and both of which are beyond the backtracking values, the station cannot be constructed, and assignments must be carried out on the stations of the other side (Rule 4). Regarding the left side stations of the third position, T9 and T10 tasks are triggered by task T7. As T9 and T10 are right side tasks, they are not assigned. After the assignment of the left side stations, the third position on the left and right sides are assigned alternatively.

When assigning the fourth position, it is carried out as a right side assignment; however, after assigning the T13 task, only the T12 task can be assigned on the left side. Thus, the fourth position task assignment can be completed by assigning the tasks of the left side stations.

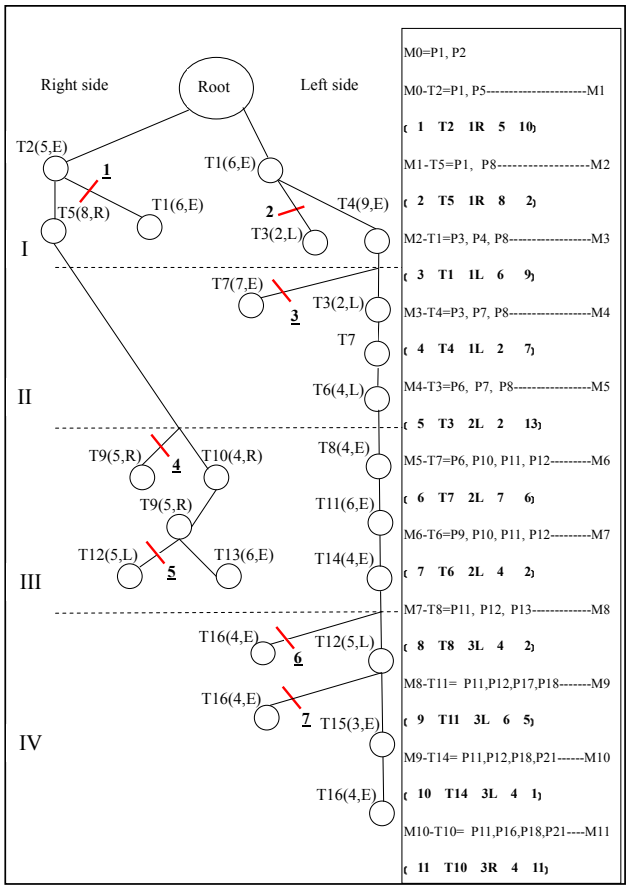


Fig. 11 Computational steps- task assignment completed

## IV. Comparative Analysis

This paper applies the proposed algorithm to solve the problems of the two-sided assembly line for comparison with other two algorithms, including ACO [12], and a genetic algorithm [1].

### *A. Comparison with ACO*

The computational results are almost identical, as both methods assign the task based on the degree of relationship (immediate, common, and sharing relationships). The slight differences of task assignment are mainly due to consideration of the task ending time; hence, the execution orders of tasks in the stations are different. If task assignment rules are not included, backtracking operation is conducted twice; however, it requires only one backtracking operation by applying the proposed method.

### *B. Comparison with the Genetic Algorithm*

As the computational results suggest, although the task arrangement in the station is different, the relationship degree of the task is considerably high (almost connected by immediate and sharing relationship). Hence, Rule 1 that is applied as the basis for task assignment according to the degree of relationship can reduce backtracking and idle time. The proposed method in this test does not produce any backtracking operation.

TABLEI
Summary of Comparison Results

| | | Position 1 | | Position 2 | | Position 3 | |
|---|---|---|---|---|---|---|---|
| | | Right side | Left side | Right side | Left side | Right side | Left side |
| 1 | The proposed method | T1, T3, T5, T4, T8 | T2, T7, T6 | T9, T12, T14 | T10, T11, T13 | | |
| | ACO | T2, T4, T5, T8 | T1, T3, T6, T7 | T12, T9, T14 | T11, T10, T13 | | |
| 2 | The proposed method | T3, T2 | T1, T4 | T5, T8 | T6, T9, T11 | T12 | T7, T10 |
| | Genetic algorithms | T2, T5 | T1, T4, | T8, T9 | T3, T6 | T11, T12 | T7, T10 |

## V. Conclusion and Suggestion

The proposed method can effectively solve the two-sided assembly line balancing problem through efficient task assignment and avoiding unnecessary backtracking operations. This study used petri net combined with the branch-and-bound method to assign tasks according to the degree of relationship between branch tasks and upper layer tasks (e.g., immediate, sharing, common, and no relationships). Moreover, it established task assignment rules to select the appropriate tasks for assignment, thus effectively reducing backtracking operation. If the rules and triggering mechanism can be modified for future branch task selection, for example, the branch task assignments can be better defined (marked by colors or tasks of lower relationship degrees are deleted in advance) to strengthen the branching ability, better computational efficiency can be achieved.

## Acknowledgment

This research was supported in part by the National Science Council, Taiwan, Republic of China, under Grant No. NSC102-2221-E-214-050. The authors would also like to thank two anonymous referees for their constructive and helpful comments.

## References

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

[2] Ugur O., and Bilal T. (2009),Multiple-criteria decision-making in two-sided assembly line balancing a goal programming and a fuzzy goal programming models, *Computers & Operations Research*, vol. 36, pp. 1955–1965.

[3] Hu X. F., Wu E. F., Bao J. S., and Jin Y. A. (2010), A branch-and-bound algorithm to minimize the line length of a two-sided assembly line, *European Journal of Operational Research*, vol. 206, pp. 703–707.

[4] Bartholdi J.J.(1993),Balancing two-sided assembly lines – A case study, *International Journal of Production Research*, vol. 31,pp. 2447–2461.

[5] Lee T. K., Kim Y., and Kim Y. K.(2001), Two-sided assembly line balancing to maximize work relatedness and slackness, *Computers & Industrial Engineering*, vol. 40, pp. 273–292.

[6] Scholl A., and Becker C. (2006), State-of-the-art exact and heuristic solution procedures for simple assembly line balancing, *European Journal of Operational Research*, vol. 168, pp. 666–693.

[7] Baykasoglu A. and Dereli T. (2008), Two-sided assembly line balancing using an ant colony-based heuristic, I*nternational Journal of Advanced Manufacturing Technology*, vol. 36, pp. 582–588.

[8] Klein R., and Scholl A.(1996), Maximizing the production rate in simple assembly line balancing – A branch-and-bound procedure, *European Journal of Operational Research*, vol. 91, pp. 367–385.

[9] Wu E.F., Jin Y., Bao J. S., and Hu X.F.(2008), A branch-and-bound algorithm for two sided assembly line balancing, *International Journal of Advanced Manufacturing Technology*, vol. 39, pp. 1009–1015.

[10] Ozcan K. (2011), Firing sequences backward algorithm for simple assembly line balancing problem of type 1, *Computers & Industrial Engineering*, vol. 60, pp.830–839.

[11] Ozcan U., and Toklu B.(2009), Multiple-criteria decision-making in two-sided assembly line balancing: A goal programming and a fuzzy goal programming models, *Computers & Operations Research*, vol. 36, pp. 1955–1965.

[12] Simaria A. S., and Vilarinho P. M. (2009), 2-ANTBAL – An ant colony optimization algorithm for balancing two-sided assembly lines, *Computers & Industrial Engineering*, vol. 56, pp. 489–506.