Meta-heuristic algorithm to solve two-sided assembly line balancing problems

A D Wirawan and A Maruf*
Faculty of Industrial Technology, Institut Teknologi Bandung, Bandung 40132, Indonesia

Email: ariodean.wirawan@gmail.com and anas@lspitb.org

Abstract. Two-sided assembly line is a set of sequential workstations where task operations can be performed at two sides of the line. This type of line is commonly used for the assembly of large-sized products: cars, buses, and trucks. This paper propose a Decoding Algorithm with Teaching-Learning Based Optimization (TLBO), a recently developed nature-inspired search method to solve the two-sided assembly line balancing problem (TALBP). The algorithm aims to minimize the number of mated-workstations for the given cycle time without violating the synchronization constraints. The correlation between the input parameters and the emergence point of objective function value is tested using scenarios generated by design of experiments. A two-sided assembly line operated in an Indonesia’s multinational manufacturing company is considered as the object of this paper. The result of the proposed algorithm shows reduction of workstations and indicates that there is negative correlation between the emergence point of objective function value and the size of population used.

1. Introduction
Assembly line balancing problems are generally divided into two categories: simple assembly line balancing problem (SALBP) and general assembly line balancing problem (GALBP). GALBP possesses more complexity than SALBP [1] due to relaxation applied towards several assumptions used in SALBP. The relaxed assumptions also lead GALBP to have better ability to represent the real-life problem situations.

Two-sided assembly line balancing problem (TALBP) is categorized as one of GALBPs. Two-sided assembly line is a set of sequential workstations where task operations can be performed in two sides of the line [2]. Two-sided assembly line has unique characteristics which make the tasks are grouped and placed along the assembly line based on their orientation (Right, Left or Either). Moreover, two-sided assembly line also consists of tasks which have to be performed by two operators whom opposing each other at two mated-workstations (synchronization constraints). Figure 1 displays the illustration of two-sided assembly line.

*Corresponding author. Tel.: +62 22 2506449; fax.: +62 22 2506449
Common TALBP goal is to minimize the number of workstations utilized at the assembly line while taking two-sided assembly line’s characteristics into account. Furthermore, other magnitude, for example the level of workload smoothness, can be considered as the attribute of the TALBP objective function. This circumstance are led by, amongs others, the different measurement units between magnitudes and the possible event where less workstations does not necessarily mean better workload smoothness.

A method which is able to solve two-sided assembly line balancing problems covering the synchronization constraints is proposed in this paper. The meta-heuristic algorithm that consists random factor generation is prefered to reduce the necessity to check every single feasible solution alternatives generated, which most likely need huge amount of computation time. The notable novelty of the proposed algorithm is the step that enable two corresponding tasks to be performed by workers at mating stations without violating any of these constraints: precedence graph and cycle time.

This paper has 5 sections which are initialized by the introduction in Section 1. The problem case overview can be found at Section 2 while complete description of the proposed algorithm is placed in Section 3. Then, Section 4 and Section 5 contains numerical example and paper’s conclusion, respectively.

2. Problem Description
An Indonesian multi-national manufacturing company utilizing two-sided assembly lines at its production floors acts as the object of this paper’s study case. The company’s management do not apply any mechanisms which are proficient to solve two-sided assembly line balancing problems systematically. Assembly line reconfigurations are held three up to five times on annual basis to anticipate the demand fluctuation, mainly because of demand increases, and the introduction of new model of products.

The complete description of this company’s assembly lines based on Boysen classification [1] are served in the following Table 1:

| Sub-category | Classification |
|--------------|----------------|
| Product specific ($\alpha_1$) | Assembly line are dedicated only for the assembly of single-model product. |
| Structure ($\alpha_2$) | Assembly line are available for every acyclic precedence graph structure. |
Table 2 Assembly Line’s Classification (cont.)

| Category: Precedence graph ($\alpha$) |
|--------------------------------------|
| **Sub-category**                      | **Classification** |
| Processing time ($\alpha_1$)          | Time required for assembly operations are static and deterministic. |
| Sequence-dependent task-time increments ($\alpha_2$) | Individual task-time increments caused by the dependencies between sequential task operations are not included. |
| Assignment restrictions ($\alpha_3$)  | Several task operations are required to be assigned at the same workstations. |
| Processing alternatives ($\alpha_4$)  | Alternative assembly operations processes or procedures are not considered. |

| Category: Station ($\beta$) |
|----------------------------|
| **Sub-category** | **Classification** |
| Movement of workpiece ($\beta_1$) | Conveyor belts with global cycle time are installed with the assembly line. |
| Line layout ($\beta_2$) | Workstations are arranged in order to form a serial line. |
| Parallelization ($\beta_3$) | Tasks operations are performed paralelly by operators at mated-stations on both sides of the assembly line. |
| Resource assignment ($\beta_4$) | Resources are not explicitly considered to do specific task operations at the assembly line. |
| Station-dependent time increment ($\beta_5$) | Workstation-time increments caused by unproductive activities are not included. |
| Additional configuration aspects ($\beta_6$) | Any additional configuration aspects are not considered at the assembly line. |

| Category: Objective ($\gamma$) |
|-------------------------------|
| The assembly line balancing process aims to minimize number of workstations with predetermined global cycle time. |

The assembly line currently consists of 172 tasks which spreads over 62 workstations with smoothness index value of 0.005 and global cycle time value of 22 seconds. This paper will propose algorithm that is able to minimize number of workstations without any violation on cycle time and synchronization constraints.

3. Proposed Algorithm
In this paper, Decoding Algorithm with Teaching-Learning Based Optimization (TLBO) is proposed to solve the particular two-sided assembly line balancing problem. Decoding Algorithm with TLBO [3] is population based algorithm which includes task assignment mechanism that take synchronization constraints into account. This algorithm has random factor to create priority assortment between task operations to avoid analytical method-like mechanism obligating to check every feasible solutions. This means less computational effort even though it costs certainty of achieving globally optimum solution.

Decoding Algorithm with TLBO consist of two main parts: task assignment part and priority assortment permutation through iterations consisting learning and teaching phases. The task assignment part is the one proposed by Tuncel (2014) that has been enriched with mechanism to cover synchronization constraints (Purnomo, et al., 2013). This part, which is called Decoding Algorithm, has random number assigned to every tasks performed in particular assembly activity acting as their own assignment priority number. The priority numbers are generated with following Equation (1):

$$x_i = 1 + (rand[0,1] + (UB - 1))$$

(1)
where UB is the upper bound or maximum number of task operations included in an assembly activity reviewed (n).

The list of priority numbers of every task are combined into a set of vector called priority list represented by Equation (2) as followed:

\[ PL = \{ x_i(1), x_i(2), \ldots, x_i(n) \} \]  \hspace{1cm} (2)

Terminologies and steps of Decoding Algorithm are showed as below:

- \( C \): cycle time (second)
- \( AT(.) \): set of assignable tasks
- \( IP_i(.) \): set of immediate predecessor of task-\( i \)
- \( ST_i(.) \): start time of task-\( i \)
- \( FT_i(.) \): finish time of task-\( i \)
- \( d \): task direction or orientation (Left, Right or Either)
- \( r \): index for task with Right orientation
- \( Sr \): cumulative station time with Right orientation
- \( l \): index for task with Left orientation
- \( Sl \): cumulative station time with Left orientation

i. **Step 1**
   Set \( r = 1, \ l = 1, \ Sr = 0, \ Sl = 0 \). Go to Step 2.

ii. **Step 2**
   Define \( AT(.) \). If \( AT(.) \) is not consisting any tasks, then go to Step 6. Otherwise, go to Step 3.

iii. **Step 3**
   Sort members of \( AT(.) \) based on the priority list \( (PL) \) in decreasing order and go to Step 4.

iv. **Step 4**
   Remove the first priority task-\( h \) from \( AT(.) \) and go to Step 5.

v. **Step 5**
   Assign the first priority task-\( h \) from \( AT(.) \) with following these guidelines:

   a. If \( d_h = R, \ t_h + Sr \leq C \), and \( t_h + FT_a \leq C \) \( (FT_a = \max\{FT_p| p \in IP_h\} \) which has been assigned at other workstations) then assigned task-\( h \) at workstation-\( r \) and set \( ST_h = \max\{Sr, FT_a\} \), \( FT_h = t_h + ST_h \), \( Sr = FT_h \), go to Step 2. Otherwise, set
      \[
      r = r + 1 \\
      Sr = 0 \\
      l = l \\
      Sl = Sl \\
      \]
      and go to Step 2.

   b. If \( d_h = L, \ t_h + Sl \leq C \), dan \( t_h + FT_a \leq C \) \( (FT_a = \max\{FT_p| p \in IP_h\} \) which has been assigned at other workstation) then assigned task-\( h \) at workstation-\( l \) and set \( ST_h = \max\{Sl, FT_a\} \), \( FT_h = t_h + ST_h \), \( Sl = FT_h \), go to Step 2. Otherwise, set
      \[
      r = r \\
      Sr = Sr \\
      l = l + 1 \\
      Sl = 0 \\
      \]
and go to Step 2.  
c. If \( d_h = E \), then generate a random real number (\( rn \)) between 0 (zero) and 1 (one) for the particular task using uniform distribution. If \( rn < 0.5 \) then go to Step 5a. If \( rn > 0.5 \) then go to Step 5b. Step 5c is available for tasks with Either orientation and without any connection with other tasks (not associated with synchronization constraints).  
d. If task-\( h \) has connection with its mating task (associated with synchronization constraints), let it be task-\( i \), where \( d_h = R \) and \( d_i = L \) or the opposite, then:  
\begin{itemize}
  \item \( IP_h = IP_i \), so that \( FT_a = \max \{ FT_p | p \in IP_h \} , \max \{ FT_q | q \in IP_i \} \).
  \item \( r = l = \max \{ r, l \} \).
  \item \( ST_h = ST_l = \max \{ ST_h , ST_l \} \).
\end{itemize}

The value \( ST_h \) and \( ST_l \) are obtained from procedure identical with the one of Step 5a and Step 5b which is computed paralellly. The results are temporary and will be compared each other to determine the permanent values of both \( ST_h \) and \( ST_l \).

For two connected tasks, define \( d_h = L \), \( d_i = R \). If \( t_h + S_l > C \) or \( t_i + S_r > C \) or \( t_h + S_r > C \) or \( t_i + S_l > C \) or \( t_h + FT_a > C \) or \( t_i + FT_a > C \), then  
\begin{align*}
  r &= r + 1 \\
  S_r &= 0 \\
  l &= l + 1 \\
  S_l &= 0
\end{align*}

and go to Step 2.  

vi. Step 6

Calculate the objective function value of the algorithm after all tasks are assigned.  

Decoding Algorithm aims to minimize number of workstations at assembly line. Smoothness index value of the configuration proposed as the result from the algorithm’s computation is considered as attribute but not included into the objective function. Smoothness index is calculated using this following Equation (15) and Equation (16):

\begin{align}
  \text{Minimize } Z &= T = \sum_{k=1}^{K} I_k \\
  C_b &= \sum_{k=1}^{K} \left[ \left( \frac{I_k}{T} \right) - \left( \frac{1}{N} \right) \right]^2
\end{align}

where
\begin{itemize}
  \item \( T \): total idle time for all workstations
  \item \( C_b \): smoothness index value
  \item \( I_k \): idle time at workstation-\( k \)
  \item \( C \): cycle time
\end{itemize}

Smaller smoothness index value indicates better level of line balance. When an assembly line has its smoothness index reaching 0.00 (zero) value, it means that the line is perfectly smooth or balanced in terms of workload level.

Decoding Algorithm are both required to generate initial solution to start TLBO and to perform the iteration of TLBO consisting learning phase and also teaching phase. TLBO is a method which is able to solve big scale optimization problem with less computation effort and high consistency [4]. TLBO is inspired by the process of knowledge transfer from teacher to students (learners) and also the learning
process of students to attain knowledge from their peers. Rao (2012) define two basic philosophies of the algorithm as follow:
1. The impact of teacher on learner’s output in the form of better result (grade or score).
2. The impact of interaction between learners to gain better result.

Teacher is defined as the individual with the best score on particular design variable, which is analogous with subject in class. This teacher can increase its learners’ grade to the point of its level. The teaching phase represents the learning method where learners get their knowledge from the teacher. At this phase, teacher tries to make the class mean \( M_i \) to move towards its grade \( M_{new} \). Following are the computation process of teaching phase:

\[
\text{Difference}_\text{Mean}_i = r_i(M_{new} - T_F M_i)
\]  \hspace{1cm} (17)

\( T_F \) : teaching factor  
\( r_i \) : random real number between \([0,1]\)

\[
T_F = round[1 + rand(0,1)(2 - 1)]
\]  \hspace{1cm} (18)

Existing solution \( X_{old,i} \) is modified using Equation (18) below:

\[
X_{new,i} = X_{old,i} + \text{Difference}_\text{Mean}_i
\]  \hspace{1cm} (19)

The value of the random number \( r_i \) varies for each solution alternative.

Meanwhile, learning phase is marked with learning process through interactions between two learners. At this phase, two non-teacher population members are chosen randomly to perform alteration on knowledge of other members (including teacher). If \( X_i \) acting as the first learner tries to improve its grade by interaction with \( X_{ii} \) as the second learner, whose grade is bigger than \( X_i \)'s, then the grade of \( X_i \) will move toward the grade of \( X_{ii} \). The opposite scenario also can happen when \( X_i \) has better result than \( X_{ii} \).

\[
\begin{align*}
X_{new,i} &= X_{old,i} + r_i(X_{ii} - X_i) & \text{if } f(X_i) < f(X_{ii}) \\
X_{new,i} &= X_{old,i} + r_i(X_i - X_{ii}) & \text{if } f(X_{ii}) < f(X_i)
\end{align*}
\]  \hspace{1cm} (20) \hspace{1cm} (21)

These Equation is applied to any objective function type without any distinction if its goal is to minimize or to maximize. It does not matter because of the meta-heuristic characteristic of the algorithm.

TLBO iteration consisting teaching and learning phase is performed repeatedly until termination condition is completed. Tuncel (2014) developed TLBO application steps based on the one of Rao (2012). These are the TLBO steps of Tuncel (2014):

i. Step 1
Define the optimization problem and initiate the optimization parameter such as number of population, number of iteration, and number of variable design. Population is alternative solutions generated in order to be included at both teaching phase and learning phase in every iteration. Design variable is the decision criteria of the objective function from the particular optimization problem.
ii. **Step 2**
Generate numbers of alternative solutions randomly synchronized with the number of population defined for every design variable. Evaluate the population’s mean value for every design variable.

iii. **Step 3**
Based on the objective function value of each individual inside the population, choose one best learner according to the respective design variable’s criteria. Calculate the population’s mean value for every design variable.

iv. **Step 4**
Evaluate the difference between the population’s mean value and the teacher’s value using Equation (17) and Equation (18).

v. **Step 5**
Update the value of every individual with teacher’s helps based on Equation (19).

vi. **Step 6**
Update the value of every individual through mutual interaction among learners. Use Equation (20) or Equation (21)

vii. **Step 7**
Repeat from Step 3 to Step 6 until the termination condition is completed.

4. **Numerical Example**
The computation process of Decoding Algorithm with TLBO is illustrated in this Figure 2 below.
Decoding Algorithm needs to generate sets of priority list so that initial solutions are obtained. The initial solutions are created from groups of assembly line’s configuration whose task’s assignments are done based on the priority list and precedence graph constraints. The best configuration will receive the teacher status for the next phase. During teaching phase, Equation (18) is used to modify the current priority lists’ value and the resulting new objective function values (OFVs) will be examined to determine whether the current solutions should be renewed or not based on the comparisons of assembly lines’ performance. Therefore, it is possible to replace current teacher with another population member after the execution of teaching phase prior to enter the learning phase. This phase uses Equation (20) or Equation (21) to update the solutions which also based on performance’s comparison. One iteration, except the initial generation, consists of two phases.

Here is Table 3 the fraction of actual data that will be used for the numerical example:

| Task ID Number | Predecessor | Mating Task | Standard Time (second) | Orientation |
|----------------|-------------|-------------|------------------------|-------------|
| 1              | Dummy       |             |                        |             |
| 2              | 1           | 4           | 10,130                 | Left        |
| 3              | 4           |             | 1,806                  | Right       |
| 4              | 1           | 2           | 3,431                  | Right       |
| 5              | 4           | 6           | 6,287                  | Left        |
| 6              | 4           | 5           | 1,381                  | Right       |
| 7              | 6           |             | 3,021                  | Left        |
| 8              | 7,9         |             | 1,128                  | Left        |
| 9              | 4           |             | 13,723                 | Left        |
| 10             | 4           |             | 1,117                  | Left        |
| 11             | 4           |             | 3,420                  | Left        |
| 12             | 11          |             | 3,013                  | Right       |
| 13             | 11          |             | 3,292                  | Right       |
| 14             | 13          |             | 3,032                  | Right       |
| 15             | 4           |             | 0,897                  | Right       |

This fraction is meant to be used as the data for the numerical example illustration. Before performing any task assignment operations, some computation parameters like precedence graph, cycle time, standard times and termination criterion should be defined. In this paper, termination criterion is met when a particular population member is considered as teacher for number of iterations performed without any changing of its objective function value. Following are the starting variable value for the numerical example:

- $r = 1$
- $l = 1$
- $Sr = 0.00$ second
- $Sl = 0.00$ second
The first available tasks are Task #2 with Task #4 which are mating stations obliged to be assigned at the same time with identical start time (see Step 5d). Therefore, for the next assignment the variable value are changed due to those previous tasks’ assignments. Below in Table 4 is shown the next available tasks with their respective priority values.

\textbf{Table 4 Available Tasks Set #1}

| AT (.) | Priority Value | Priority Order |
|--------|----------------|----------------|
| 3      | 1.98           | 4              |
| 5      | 13.51          | 2              |
| 6      | 13.51          | 2              |
| 9      | 21.38          | 1              |
| 10     | -24.33         | 6              |
| 11     | -1.45          | 5              |
| 15     | 2.01           | 3              |

i. Step 1
The current variables are \( r = 1; l = 1; S_r = 3,431; S_l = 10,130 \).

ii. Step 2
The next assignable tasks set is \( AT = \{3,5,6,9,10,11,15\} \).

iii. Step 3
Sort the assignable tasks set based on priority order. The sorted set is \( AT = \{9,5,6,15,3,11,10\} \). Thus, Task #9 is chosen.

iv. Step 4
Remove Task #9 from \( AT \).

v. Step 5b
It is started with \( d_9 = L; \)
\( r_9 + FT_4 = 13,723 + 3,431 \leq 22 \); due to \( r_9 + S_l = 13,723 + 10,130 > 22 \); so \( l = l + 1 = 2; S_l = 0 \). Then \( r_9 + S_l = 13,723 + 0 \leq 22 \); which makes \( ST_9 = 0; FT_9 = 13,723; S_l = 13,723 \). Go to Step 2.

\textbf{Table 5 Available Tasks Set #2}

| AT (.) | Priority Value | Priority Order |
|--------|----------------|----------------|
| 3      | 1.98           | 4              |
| 5      | 13.51          | 1              |
| 6      | 13.51          | 1              |
| 10     | -24.33         | 6              |
| 11     | -1.45          | 5              |
| 15     | 2.01           | 3              |
| 26     | 5.23           | 2              |
| 27     | 5.23           | 2              |

vi. Step 2
The current variables are \( r = 1; l = 2; S_r = 3,431; S_l = 13,723 \). The next assignable tasks set is \( AT = \{3,5,6,10,11,15,26,27\} \).

vii. Step 3
Sorted set is \( AT = \{5,6,26,27,15,3,11,10\} \). Thus, Task #5 and Task #6, which are mating stations, are chosen.
viii. Step 4
Remove Task #5 and Task #6 from AT.

ix. Step 5d
It is started with the connected \( ds = L; \) \( d0 = R; \) because \( l > r \), then the workstation index is synchronized as \( r = l = 2 \). Next, \( t5 + Sl = 6,287 + 13,723 \leq 22; \) \( t6 + Sr = 1,381 + 0 \leq 22; \) \( t5 + FT9 = 6,287 + 13,723 \leq 22; \) \( t6 + FT9 = 1,381 + 13,723 \leq 22 \).

Hence, \( ST5 = \max \{ Sl, FT9 \} = 13,723; \) \( ST6 = \max \{ Sr, FT9 \} = 13,723 \); which implies that \( ST5 = ST6 = 13,723; \) and \( FT5 = ST5 + t5 = 13,723 + 6,287 = 20,010; \) \( FT6 = ST6 + t6 = 13,723 + 1,381 = 15,104; \) \( Sl = 20,010; \) \( Sr = 15,104 \).

Go to Step 2.

Continue until all tasks are assigned according to the algorithm steps.

5. Result and Conclusion
The computations of Decoding Algorithm with TLBO using several different parameter sets yield the best two-sided assembly line configuration consisting 59 stations with smoothness index value of 0,01276. Table 6 shows the parameter sets used and their corresponding results:

Table 6 Results’ Comparison between Parameter Sets

| Population members | Iteration termination | Initial generation time (second) | Initial teaching phase time (second) | Initial learning phase time (second) | 6th iteration | Objective function value |
|--------------------|-----------------------|---------------------------------|-------------------------------------|-------------------------------------|---------------|--------------------------|
| 10                 | 1,000                 | 5.98                            | 9.11                                | 11.34                               | 196           | 368 753 1,363 3,302     |
| 100                | 250                   | 80.8                            | 124.79                              | 126.71                              | 32            | 34 260 - -             |
| 250                | 25                    | 226.78                          | 247.45                              | 224.95                              | 4             | 15 15 24 47           |

Performed using Macro VBA additional installation on Microsoft Excel, the Decoding Algorithm with TLBO computation’s results indicate that the size of population member is negatively correlated with the emergence points of OFV. However, it will be more representing to compare each population number set based on their respective total computation time. Moreover, it is important to be noted that the time displayed are only for computation time which exclude the time required to perform the visualization process of the final configuration proposed. A robust design of experiment must be proposed in order to show the comprehensive behavior of Decoding Algorithm with TLBO regarding the impact of its parameters: population numbers, random number generator, and the data size (number of tasks) so that the algorithm is eligible to be compared fairly with other solving algorithm alternatives. The development of better computing software or application is also required to perform the Decoding Algorithm with TLBO that is easier to be verified and validated.

References
[1] Boysen, N., Fliedner, M. & Scholl, A., 2007. Classification for Assembly line balancing problems. European Journal of Operational Research, 183(2), pp. 674.
[2] Purnomo, H. D., Wee, H.-M. & Rau, H., 2013. Two-sided assembly lines balancing with assignment restrictions. Mathematical and Computer Modelling, Volume 57, pp. 189-199.
[3] Tuncel, G. & Aydin, D., 2014. Two-sided assembly line balancing using teaching-learning based optimization. Computers & Industrial Engineering, Volume 74, pp. 291-299.
[4] Rao, R., Savsani, V. J. & Vakharia, D. P., 2012. Teaching-learning based optimization: An optimization method for continuous non-linear large scale problems. Information Sciences, Volume 183, pp. 1-15.
[5] Agpak, K., 2010. An approach to find task sequence for re-balancing of assembly lines. Assembly Automation, 30(4), pp. 378-387.
which model to use when?. *International Journal of Production Economics*, 111(2), p. 509

[6] Gamberini, R., Grassi, A. & Rimini, B., 2006. A new multi-objective heuristic algorithm for solving the stochastic assembly line re-balancing problem. *International Journal of Production Economics*, Volume 102, p. 226.

[7] Kim, Y. K., Kim, Y. & Kim, Y. J., 2000. Two-sided assembly line balancing: A genetic algorithm approach. *Production Planning and Control*, 11(1), pp. 44-53.

[8] Ozbakir, L. & Tapkan, P., 2011. Bee colony intelligence in zone constrained two-sided assembly line balancing problem. *Expert Systems with Applications*, Volume 38, pp. 11947-11957.

[9] Rao, R. V., Savsani, V. J. & Vakharia, D. P., 2011. Teaching-learning based optimization: A novel method for constrained mechanical design optimization problems. *Information Sciences*, Volume 43, pp. 303-315.