The type E simple assembly line balancing problem: A mixed integer linear programming formulation

Rasul Esmaeilbeigi, Bahman Naderi, Parisa Charkhgard

PII: S0305-0548(15)00144-6
DOI: http://dx.doi.org/10.1016/j.cor.2015.05.017
Reference: CAOR3799

To appear in: *Computers & Operations Research*

Received date: 16 September 2014
Revised date: 19 January 2015
Accepted date: 30 May 2015

The type E simple assembly line balancing problem: A mixed integer linear programming formulation

Rasul Esmaeilbeigi[1], Bahman Naderi [*,1], and Parisa Charkhgard[2]

[1] *Department of Industrial Engineering, Faculty of Engineering, Kharazmi University, Tehran, Iran*

[2] *Department of Industrial Engineering, Sharif University of Technology, Tehran, Iran*

June 4, 2015

Abstract

Although the simple assembly line balancing problem (SALBP) is the topic of many studies, typically they either consider minimizing the number of stations for a given cycle time (called type one), or minimizing the cycle time for a given number of stations (called type two). Rarely, type E of the problem is considered. In the type E, cycle time and number of stations are both decision variables, and the objective is to maximize the line efficiency. This paper presents a mixed integer linear programming formulation for the type E simple assembly line balancing problem. Moreover, to further strengthen the presented formulation, two enhancement techniques in form of valid inequalities and auxiliary variables are proposed. As the secondary objectives of the problem, minimization of the number of stations, the cycle time, and the smoothness index are studied as well. In the case of workload smoothing, three different linearization methods are employed and compared for minimizing the smoothness index. The results of computational study on the benchmark data set demonstrate the efficacy of the improved formulation.

Keywords— Simple assembly line balancing, Workload smoothing, Linearization, Type E

1 Introduction

Over the past six decades, the topic of assembly line balancing has received considerable attention from those who either (both) study or (and) implement production planning. The assembly line balancing problem (ALBP) consists of assigning tasks to an ordered sequence of stations such that the precedence relations among the tasks are satisfied and some performance measure is optimized (Erel and Sarin [9]). The simple assembly line balancing problem (SALBP), the basic version of the problem, has been examined extensively. One can refer interested readers to surveys by Baybars [4], Ghosh and Gagnon [12], Erel and Sarin [9], Rekiek et al. [19], Scholl and Becker [22], Battaa and Dolgui [3], Saif et al. [20], and Sivasankaran and Shahabudeen [24] for details.

[*] Corresponding author. Tel.: +98 263 4551022
E-mail address: *bahman.naderi@aut.ac.ir*

These problems are commonly classified into two categories; SALBP-1 and SALBP-2. In the SALBP-1, for a given cycle time (or for a given production rate) the number of required stations is minimized. In the SALBP-2, the number of stations is given, and the objective is to minimize the cycle time (or maximize the production rate). The more generalized SALBP is called SALBP-E where both the cycle time and the number of stations are decision variables for the problem. The objective of the SALBP-E is to maximize the line efficiency by minimizing the product of the cycle time and the number of stations. Unlike the SALBP-1 and SALBP-2, this problem is rarely studied by researchers, likely because of its non-linear form and the difficulty of dealing with (see Battaa and Dolgui [3]).

Scholl and Becker [22] mention that procedures directly solving the SALBP-E are not available. Indeed, the SALBP-E is typically solved by iteratively solving several SALBP-1 or SALBP-2 instances. For example, Wei and Chao [26] propose an exact solution procedure based on solving several SALBP-2 formulations. They also present a non-linear model which combines the SALBP-1 and SALBP-2. The work of Zacharia and Nearchou [28] can be seen as the first attempt to solve the SALBP-E directly, without solving several SALBP-1 or SALBP-2 instances. They assume that the cycle time and the task processing times are triangular fuzzy numbers, and refer to this problem as the fuzzy SALBP-E (f-SALBP-E). To solve this problem, they use a meta-heuristic based on genetic algorithms. As far as we know, no exact solution procedure or mathematical formulation exists in the literature to directly solve the SALBP-E.

The SALBP is first formulated as integer linear programming formulations by Bowman [5]. He develops two different formulations for the SALBP-1. In the first formulation he considers some decision variables representing the amount of time which a task devotes to a station. To assure that task times do not split between the stations, he uses a set of binary variables to provide non-divisibility constraints. In the second formulation, he considers some decision variables representing the clock time when a task is started. Accordingly, to assure that tasks do not use the same clock time, he employs a set of binary variables and adds the non-interference constraints. Later, White [27] reformulates Bowman's first model and shows that the new model requires fewer variables and constraints. In this formulation, a set of binary variables are used which take value one if and only if a task is assigned to a specific station. Based on these variables, Patterson and Albracht [17] introduce the concept of earliest and latest stations to which a task can be assigned (a preprocessing for the formulation). They compare the new formulation with the previous ones and conclude that the new formulation consists of fewer variables and constraints. Baybars [4] presents a binary integer linear programming formulation for the SALBP-2. The formulation is a modification of White's formulation, in which the objective function is adapted. Pastor and Ferrer [16] consider a modification of the formulation presented by Patterson and Albracht [17]. They point out that in addition to the precedence relations, the earliest and latest stations for a specific task depend on the upper bound on the number of stations for the SALBP-1, or the upper bound on the cycle time for the SALBP-2. They introduce a set of valid inequalities for the SALBP-1 and SALBP-2 to impose this concept on the formulation. Using the new approach, they can solve a greater number of instances to optimality by the improved formulation.

As reviewed, one important research line in the SALBPs is to develop effective mathematical programming formulations. But, the type E of the problem has not been linearly formulated. This paper presents a mixed integer linear programming (MILP) formulation for the problem, which can be solved by standard linear solvers. Moreover, to further strengthen the presented formulation, several improvements in form of valid inequalities and auxiliary variables are proposed. The improved formulation is then capable of solving small and medium instances of the problem to optimality, within a reasonable amount of time. Furthermore, the application of different secondary objectives in the problem such as the workload smoothing are studied. To the best of our knowledge, this is the first exact method presented to solve the SALBP-E directly by an MILP formulation.

The rest of the paper is organized as follows. Section 2 formally describes the problem under study and reviews some preliminary notations. Section 3 introduces an MILP formulation for the

SALBP-E. Moreover, some secondary objectives such as the workload smoothing are considered for the formulation. In the case of workload smoothing, because the secondary objective is non-linear, three different linearization methods are employed and compared. Section 4 discusses some improvement techniques to enhance the formulation. Section 5 computationally evaluates the performance of the model on the benchmark data set. Finally, Section 6 provides some concluding remarks.

2 Problem description and notations

The SALBP-E can be described as follows. A single product (so called *single-model*) is manufactured on a serial line. The total amount of work for manufacturing the product is partitioned into a set of elementary operations named tasks. Due to technological and organizational conditions, there exist some precedence relations among the tasks, which constitute edges (arcs) of a *precedence graph*. In the precedence graph, only direct predecessors (or successors) of each task (node) are shown (the set of all predecessors and successors of a task can be obtained by computing the transitive closure of the precedence graph). Performing each task takes a *task time*, which is the weight of the task in the precedence graph. It is assumed that the task times are deterministic and integer values. Each task must be assigned to exactly one station, but the stations can include more than one task. The total execution time of tasks assigned to a station is its *station time*. Due to continuous movement of the line, each product unit remains at each station for a fixed time span, which is called the *cycle time*. Each station time must not exceed the cycle time. In other words, the cycle time is equal to the maximal station time. The difference between the cycle time and the station time is called the *idle time* for the station. Due to integrality of the cycle time (the maximal station time), the idle times are also integer values. We denote the total idle time for the stations by I, the cycle time by c, and the number of stations by m. The required *line capacity*, denoted by T, is defined as the product of the cycle time and the number of stations, i.e. $T = m \cdot c$. The SALBP-E seeks to find a feasible combination (m, c) so as to minimize T. Table 1 defines the notation and parameters for the SALBP-E.

A SALBP-E instance is defined by an interval $[\underline{c}, \bar{c}]$ of possible cycle times and/or an interval $[\underline{m}, \bar{m}]$ of possible numbers of stations (Scholl and Becker [22]). For example, in the SALBP-E's benchmark (see `http://alb.mansci.de`), just the bounds on the number of stations (intervals $[\underline{m}, \bar{m}]$) are given. In fact, whenever the bounds on one of the variables m and c are given, bounds on the other variable can be calculated theoretically. As is shown by Proposition 1, for a given interval $[\underline{m}, \bar{m}]$, the lower bound and upper bound on the cycle time can be calculated theoretically as a function of $\bar{m}$ and $\underline{m}$, respectively. In the case that interval $[\underline{c}, \bar{c}]$ is given, one can simply set $\underline{m} = \lceil t_{sum}/\bar{c} \rceil$ and $\bar{m} = n$.

Proposition 1. *In the SALBP-E, $\underline{c} = f(\bar{m})$ and $\bar{c} = g(\underline{m})$ are valid bounds on the optimal value of the cycle time, where*

$$f(m) := \max\{t_{max}, \lceil \frac{t_{sum}}{m} \rceil\} \tag{1}$$

$$g(m) := \begin{cases} \max\{t_{max}, \lfloor \frac{2 \cdot (t_{sum} - 1)}{m} \rfloor\} & \textit{If } m \textit{ is even,} \\ \max\{t_{max}, \lfloor \frac{2 \cdot t_{sum}}{m+1} \rfloor\} & \textit{If } m \textit{ is odd.} \end{cases} \tag{2}$$

Proof. Hackman et al. [13] prove that for a SALBP-2 instance, with m stations given, $\underline{c} = f(m)$ and $\bar{c} = g(m)$ are valid bounds on the cycle time, if execution times are integer. It is obvious that a SALBP-E can be considered as $\bar{m} - \underline{m} + 1$ SALBP-2 instances. Since $f(m)$ and $g(m)$ are non-increasing functions

Table 1: The notation and parameters for the SALBP-E

Notation	Definition
n	The number of tasks
V	Set of tasks, i.e., $V = 1, 2, ..., n$
i, j	Indices for the tasks
k	Index for the stations
t_i	The execution time for task $i \in V$
t_{min}	The minimum of the task times, i.e., $t_{min} = \min_{i \in V}(t_i)$
t_{max}	The maximum of the task times, i.e., $t_{max} = \max_{i \in V}(t_i)$
t_{sum}	The sum of the task times, i.e., $t_{sum} = \sum_{i \in V} t_i$
$\underline{c}$	The lower bound on the cycle time
$\bar{c}$	The upper bound on the cycle time
$\underline{m}$	The lower bound on the number of stations
$\bar{m}$	The upper bound on the number of stations
KD	The set of definite stations, i.e., $KD = \{1, ..., \underline{m}\}$
KP	The set of probable stations, i.e., $KP = \{\underline{m} + 1, ..., \bar{m}\}$
K	The set of all possible stations, i.e., $K = KD \cup KP$
R	The set of direct precedence relations
P_i	The set of all (direct and indirect) predecessors of task $i \in V$
F_i	The set of all (direct and indirect) successors of task $i \in V$
t_i^p	The execution time for all predecessors of task $i \in V$, i.e., $t_i^P = \sum_{j \in P_i} t_j$
t_i^F	The execution time for all successors of task $i \in V$, i.e., $t_i^F = \sum_{j \in F_i} t_j$
E_i	The earliest station for task $i \in V$, i.e., $E_i = \lceil \frac{t_i + t_i^p}{\bar{c}} \rceil$
L_i	The latest station for task $i \in V$, i.e., $L_i = \bar{m} + 1 - \lceil \frac{t_i + t_i^F}{\bar{c}} \rceil$
FS_i	The set of stations to which task $i \in V$ is feasibly assignable, i.e., $FS_i = \{E_i, E_i + 1, ..., L_i\}$
FT_k	The set of tasks are feasibly assignable to station $k \in K$, i.e., $FT_k = \{i \in V \mid k \in FS_i\}$

of m, $f(\bar{m})$ and $g(\underline{m})$ give the smallest lower bound and the largest upper bound in these instances, respectively. So, they can be considered as valid bounds for the cycle time in the SALBP-E. $\square$

The *line efficiency*, denoted by η, is defined as

$$\eta := \frac{t_{sum}}{T} \times 100\% \tag{3}$$

As earlier mentioned, the objective of the SALBP-E is to minimize the required line capacity T. Since t_{sum} is a constant value, maximizing η is equivalent to minimizing T. Thus, the objective of the SALBP-E is also maximization of the line efficiency defined by equation (3). On the other hand, in the SALBP, the relation $T = I + t_{sum}$ holds (see Scholl [21]). Therefore, to maximize the line efficiency, one can minimize the total idle time. Later, in Section 3, by the use of this point, we formulate the SALBP-E as an MILP formulation.

An illustrative example is presented to further explain the SALBP-E. The example is problem "Jackson" with $[\underline{m}, \bar{m}] = [3, 7]$ from the benchmark. Fig 1 shows the precedence graph with $n = 11$ tasks and the task times as the node weights.

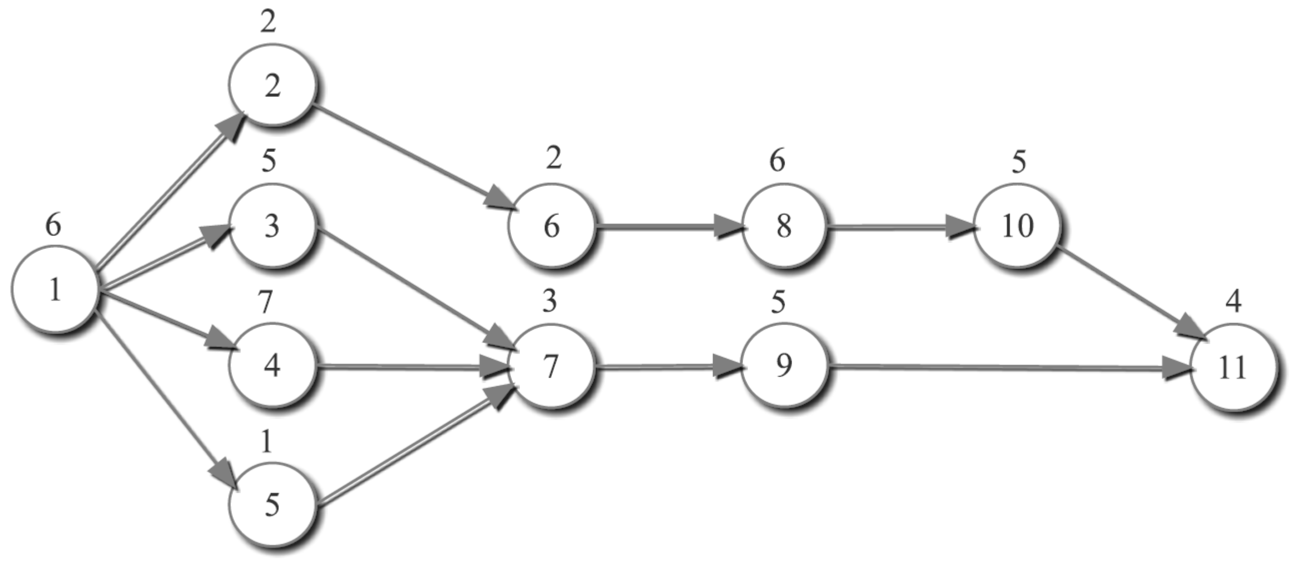

Figure 1: Precedence graph of the Jackson's problem

To solve a SALBP-E instance, the first step is to provide bounds on the number of stations or the cycle time. Since the bounds on the number of stations are given, Proposition 1 is used to calculate bounds on the cycle time. Considering $t_{sum} = 46$, the bounds are calculated as $[\underline{c}, \bar{c}] = [f(7), g(3)] = [7, 23]$.

This instance of the problem has two different optimal combinations $(m, c) = (3, 16)$ and $(m, c) = (4, 12)$ with $T = 48$. Observe that combination $(m, c) = (6, 8)$ with $T = 48$ is infeasible because for $c = 8$ at least $m = 7$ stations are required. Table 2 details four optimal solutions for the example. From the table we see that there might be more than one optimal combination of (m, c), and also more than one solution corresponding to each optimal combination. One can distinguish these optimal solutions by defining some secondary objectives. Secondary objectives are commonly employed in the assembly line balancing problems. For example, recently Scholl et al. [23] use a secondary objective for minimizing the setup times in an assembly line balancing and scheduling problem with sequence-dependent setup times. Because in the SALBP-E several optimal combinations of the cycle time and the number of stations are possible, minimizing the cycle time (maximizing the production rate) and minimizing the number of stations (minimizing the station related costs) can be considered as the secondary objectives. The reader is referred to Amen [1] for different costs connected to the number of stations.

Also, it is notable to point out that solutions to the problem may not assign equal workloads to the stations, and therefore create operating inefficiencies (see Rachamadugu and Talbot [18]). For instance, the distribution of the idle times for solutions 2 and 4 is smoother, which makes them superior to the other two solutions. To equalize the station times (to smooth the station loads), a secondary objective

Table 2: Some optimal solutions for the example

Combination	Solution	Station	Tasks	Station time	Idle time
(m,c)=(3,16)	1	1	{1,2,6,8}	16	0
		2	{3,4,5,7}	16	0
		3	{9,10,11}	14	2
	2	1	{1,2,4}	15	1
		2	{3,5,6,7,9}	16	0
		3	{8,10,11}	15	1
(m,c)=(4,12)	3	1	{1,3,5}	12	0
		2	{2,6,8}	10	2
		3	{4,10}	12	0
		4	{7,9,11}	12	0
	4	1	{1,2,5,6}	11	1
		2	{3,8}	11	1
		3	{4,10}	12	0
		4	{7,9,11}	12	0

can be used. This secondary objective can be expressed in the form of different functions (see e.g., Eswaramoorthi et al. [11] and Emde et al. [8]). One of the most famous functions is the *smoothness index (SX)*, which is defined as follows:

$$SX := \sqrt{\sum_{k=1}^{m}(c - t'_k)^2} \tag{4}$$

In this definition, t'_k is the station time for station k. Recently, for the SALBP with a given value of the number of stations, Mozdgir et al. [15] present a non-linear model in which the smoothness index is minimized as the primary objective of the model. In this paper, we minimize the smoothness index as a secondary objective for the SALBP-E. To optimally solve the problem under hand, three different linear functions are also introduced in Section 3.

We include each secondary objective and the primary objective of the problem into a single objective function. That is, by solving an MILP formulation, the maximization of the line efficiency and optimization of the secondary objective are performed simultaneously. Note that the secondary objectives can also be optimized lexicographically, i.e., by procedures that solve a sequence of single-objective optimization problems. In other words, the problem is first solved to optimize the primary objective, and then some information is transfered to a second problem with the objective of optimizing the secondary objective. This approach is not examined in this paper.

3 The SALBP-E formulation

In this section, a mixed integer linear program is developed for the problem. The decision dimension of this problem is the assignment of tasks to stations. The decision variables are defined as follows. For each task $i \in V$, and station $k \in FS_i$, the binary decision variable x_{ik} indicates whether the task is assigned to that station, that is,

$$x_{ik} := \begin{cases} 1 & \text{If task i is assigned to station k,} \\ 0 & \text{Otherwise.} \end{cases}$$

Also, for each station $k \in KP$, the binary decision variable u_k indicates whether any task is assigned to that station, that is,

$$u_k := \begin{cases} 1 & \text{If any task is assigned to station k,} \\ 0 & \text{Otherwise.} \end{cases}$$

Using these variables, the SALBP-1 formulation (F1) can be expressed as follows:

$$\begin{aligned}
\text{F1:}\ \min \quad & \sum_{k \in KP} k \cdot u_k && && (5)\\
\text{subject to:} \quad & \sum_{k \in FS_i} x_{ik} = 1 && \forall i \in V && (6)\\
& \sum_{k \in FS_i} k \cdot x_{ik} \le \sum_{k \in FS_j} k \cdot x_{jk} && \forall (i,j) \in R && (7)\\
& \sum_{i \in FT_k} t_i \cdot x_{ik} \le c && \forall k \in KD && (8)\\
& \sum_{i \in FT_k} t_i \cdot x_{ik} \le c \cdot u_k && \forall k \in KP && (9)\\
& x_{ik} \in \{0,1\} && \forall i \in V \text{ and } k \in FS_i && (10)\\
& u_k \in \{0,1\} && \forall k \in KP. && (11)
\end{aligned}$$

The objective function (5) has two important roles. First it minimizes the number of stations, by minimizing the weighted number of probable stations. Secondly, it removes any empty stations which appear between two consecutive stations, i.e., it ensures that $u_{k+1} = 0$ if $u_k = 0$. Thus, in addition to minimizing the number of stations, it breaks the symmetry in the formulation. After solving the formulation, the number of stations, m, is calculated using $m = \underline{m} + \sum_{k \in KP} u_k$. Constraint set (6) enforces each task to be assigned to exactly one station. Constraint set (7) imposes precedence relations on tasks. Constraint sets (8) and (9) ensure that station times do not exceed the given cycle time. Finally, constraint sets (10) and (11) ensure that decision variables x_{ik} and u_k are binary.

The SALBP-2 minimizes the cycle time for a given number of stations. Therefore, any variables and constraints associated with the probable stations (KP) are not used in the SALBP-2 formulation (F2). The formulation is expressed as follows:

$$\begin{aligned}
\text{F2:}\ \min\ & c && (12)\\
\text{subject to:}\quad & (6),\ (7),\ (8),\ (10),\ \text{and:} &&\\
& \underline{c} \le c \le \bar{c}. && (13)
\end{aligned}$$

In F2 objective function (12) minimizes the cycle time. Also constraints (13) are domain constraints for variable c.

As can be seen, variable c is not used in F1, as well as, variable u_k is not used in F2. However, the SALBP-E formulation (FE) includes both of these variables as well as x_{ik} variables. In fact, FE can be formulated by using these well-known variables, and a new set of continuous variables. In this case, the SALBP-E is formulated as a mixed integer linear program. We introduce for each station $k \in K$, variable δ_k as the idle time for the station. Thus, $\bar{m}$ continuous variables are defined. FE can be expressed as follows:

$$\text{FE: } \min I = \sum_{k \in K} \delta_k \tag{14}$$

subject to: (6), (7), (10), (11), (13), and:

$$\sum_{i \in FT_k} t_i \cdot x_{ik} + \delta_k = c \qquad \forall k \in KD \tag{15}$$

$$\sum_{i \in FT_k} t_i \cdot x_{ik} + \delta_k \leq c \qquad \forall k \in KP \tag{16}$$

$$\sum_{i \in FT_k} t_i \cdot x_{ik} + \delta_k \geq c + \bar{c} \cdot (u_k - 1) \qquad \forall k \in KP \tag{17}$$

$$\sum_{i \in FT_k} t_i \cdot x_{ik} + \delta_k \leq \bar{c} \cdot u_k \qquad \forall k \in KP \tag{18}$$

$$u_{k+1} \leq u_k \qquad \forall k \in KP \backslash \{\bar{m}\} \tag{19}$$

$$\delta_k \geq 0 \qquad \forall k \in K. \tag{20}$$

In FE, objective function (14) minimizes the total idle time for the stations. Let us remind that the total idle time minimization ensures efficiency maximization. Constraint sets (15), (16), (17) and (18) capture the idle times for station k. Note that the idle time of a station becomes zero if no task is assigned to that station. Constraint set (19) avoids the symmetry. Finally, constraint set (20) defines decision variables of the idle times. We know that the idle time variables can only take integer values in advance. The reason why the idle time variables are considered as continuous variables is that constraint sets (15), (16), (17) and (18) make the idle time variables take integer values even if they are relaxed. So, these variables can be safely set as continuous variables, hence the need for fewer integer variables in the formulation.

Experimental results indicate that in spite of adding more integer variables into the formulation, considering these variables as integer variables gives much better performance (see Section 5). This is because the IP solver is capable of generating general cutting planes such as the well-known *Gomory Fractional Cuts* due to integrality of the idle time variables. Thus, to exploit this ability of the IP solver, one should change the non-negativity constraints (20) to integrality constraints. This modification also indicates the integrality of the objective function. So, *objective integrality cuts* can also be generated by the solver. In other words, if the objective coefficients are all integer, then the objective value also must be an integer because the variables are required to be integer. In this case, if $\underline{I}$ is the best lower bound found so far, the objective integrality cut $\sum_{k \in K} \delta_k \geq \lceil \underline{I} \rceil$ can be generated. The interested reader is referred to Chen et al. [7] for further information on different kinds of cutting planes.

Comparing with F1, FE only requires $\bar{m}$ more continuous variables and $3 \cdot (\bar{m} - \underline{m}) - 1$ additional constraints. Comparing with F2, it just includes $\bar{m}$ additional continuous variables. As can be seen, we succeeded to formulate the SALBP-E as a linear program with a few additional variables and constraints, i.e., the number of additional variables and the number of additional constraints required is linearly bounded by the number of tasks.

It is obvious that the proposed model for FE can also be used to solve both SALBP-1 and SALBP-2. This is due to the fact that the SALBP-E is the more general version of the problem, and therefore any formulation which solves SALBP-E instances is also valid for solving the SALBP-1 and SALBP-2.

As earlier discussed in Section 2, a SALBP-E instance may include more than one optimal combination of the cycle time and the number of stations. One might be interested to minimize either the cycle time or the number of stations as a secondary objective. To this end, the following expressions can be considered as the objective function.

$$\min\ (\bar{m} - \underline{m} + 1) \cdot \sum_{k \in K} \delta_k - \sum_{k \in KP} u_k \tag{21}$$

$$\min\ (\bar{m} - \underline{m} + 1) \cdot \sum_{k \in K} \delta_k + \sum_{k \in KP} u_k \tag{22}$$

Both objective functions (21) and (22) maximize the line efficiency as the primary objective. Among optimal combinations (m, c), objective function (21) maximizes the number of stations as the secondary objective, which is equivalent to minimizing the cycle time. Also, objective function (22) minimizes the number of stations as the secondary objective. Note that these secondary objectives do not influence maximization of the line efficiency because the total idle time is an integer value (consider the division of each expression by $\bar{m} - \underline{m} + 1$). We write the objective functions in such a form to make sure that the objective function coefficients are all integers.

Another secondary objective for the SALBP-E is workload smoothing (or leveling), which can be executed by minimizing the smoothness index (see equation (4)). We write the smoothness index in terms of the idle time variables as follows:

$$SX := \sqrt{\sum_{k=1}^{m} \delta_k^2} \tag{23}$$

It is obvious that instead of minimizing the smoothness index directly, one can minimize SX^2. This is very helpful because dealing with quadratic terms $\sum_{k=1}^{m} \delta_k^2$ is much easier. We first present a property of the real numbers, which can be used to linearize these quadratic terms. Suppose that $r_1, r_2, ..., r_M$ are M real numbers. Then, the following property holds.

$$(\sum_{\ell=1}^{M} r_\ell)^2 = \sum_{\ell=1}^{M} r_\ell^2 + 2 \sum_{\ell=1}^{M} \sum_{\ell'=\ell+1}^{M} r_\ell \cdot r_{\ell'} \tag{24}$$

Property (24) can also be used in developing a theoretical upper bound for the smoothness index. This theoretical upper bound is presented by Proposition 2.

Proposition 2. *In the SALBP-E,* $u_{SX} := \max(\bar{m} \cdot t_{max}, 2 \cdot t_{sum}) - t_{sum}$ *is an upper bound on the optimal value of the smoothness index.*

Proof. Suppose that $\bar{I}$ is an upper bound on the total idle time, i.e., $\sum_{k=1}^{m} \delta_k \leq \bar{I}$. Because the idle times are non-negative integer variables, property (24) implies that $SX \leq \bar{I}$. So, any upper bound on the total idle time is an upper bound on the smoothness index as well. It is enough to show that $\max(\bar{m} \cdot t_{max}, 2 \cdot t_{sum}) - t_{sum}$ is an upper bound on the total idle time. From Proposition 1, we see that $m \cdot c \leq m \cdot \bar{c} \leq \max(\bar{m} \cdot t_{max}, 2 \cdot t_{sum})$. The assertion is proved by replacing $m \cdot c$ with $I + t_{sum}$. $\square$

Using the upper bound provided by Proposition 2, the following objective function is used for maximizing the line efficiency as the primary objective, and minimizing the smoothness index as the secondary objective.

$$\min\ (u_{SX}^2 + 1) \cdot \sum_{k \in K} \delta_k + \sum_{k \in K} \delta_k^2 \tag{25}$$

Note that due to integrality of the idle time variables, objective function (25) does not influence minimization of the total idle time. We present three different methods for linearizing objective function (25). The first method is to introduce time index variables γ_{kh} for each $k \in K$ and $h \in H = \{1, 2, ..., \bar{c} - t_{min}\}$ as follows.

$$\gamma_{kh} := \begin{cases} 1 & \text{If } \delta_k = h \ , \\ 0 & \text{Otherwise.} \end{cases}$$

Now, objective function (25) can be linearized as follows.

$$\min \ (u_{SX}^2 + 1) \cdot \sum_{k \in K} \delta_k + \sum_{k \in K} \sum_{h \in H} h^2 \cdot \gamma_{kh} \tag{26}$$

Note that constraint sets (27), (28) and (29) are also included into the model.

$$\sum_{h \in H} \gamma_{kh} \leq 1 \qquad \forall k \in K \tag{27}$$

$$\sum_{h \in H} h \cdot \gamma_{kh} \geq \delta_k \qquad \forall k \in K \tag{28}$$

$$\gamma_{kh} \in \{0, 1\} \qquad \forall k \in K \text{ and } h \in H \tag{29}$$

Constraint set (27) guarantees that for each station k, at most one variable γ_{kh} is positive (the idle time for station k can only take one unique value). When variable δ_k is positive, constraint set (28) in conjunction with constraint set (27) assures that exactly one of the variables γ_{kh} take value one. Otherwise, objective function (26) ensures that all variables γ_{kh} corresponding to station k take value zero. Constraint set (29) makes sure that decision variables γ_{kh} are binary.

The major disadvantage of this method is that it requires a great number of binary variables, i.e., $\bar{m} \cdot (\bar{c} - t_{min})$ binary variables are defined. Moreover, it includes so many non-zero elements into the coefficient matrix of the formulation. Due to this linearization, the number of non-zero elements that are included into the objective function and constraints of the model is $\bar{m} \cdot (3 \cdot (\bar{c} - t_{min}) + 1)$. In the second and third methods, we try to resolve these drawbacks.

We know that the product of two binary variables can be easily linearized. Moreover, for a binary variable d, it is obvious that $d = d^2 = d^3 = ...$. Taking this idea into account, it seems that the *binary or 0-1 representation* of integer variables δ_k can be advantageous. Let $a = \log_2^{\bar{c} - t_{min}}$. The binary representation of integer variable δ_k in terms of binary variables y_{pk} is $\delta_k = \sum_{p \in A} 2^p \cdot y_{pk}$, in which $A = \{0, 1, 2, ..., \lfloor a \rfloor\}$. Considering property (24), quadratic term δ_k^2 is represented as

$$\delta_k^2 = \sum_{p \in A} 4^p \cdot y_{pk} + \sum_{p \in A} \sum_{q \in C_p^A} 2^{(p+q+1)} \cdot y_{pk} \cdot y_{qk}$$

in which $C_p^A = \{q \in A \mid q \geq p + 1\}$. To linearize the quadratic term $y_{pk} \cdot y_{qk}$, binary variable z_{pqk} is used. After linearization, minimization of the smoothness index can be executed by the use of objective function (30) and addition of constraint sets (31), (32), (33), and (34) into the model.

$$\min \ (u_{SX}^2 + 1) \cdot \sum_{k \in K} \delta_k + \sum_{k \in K} \sum_{p \in A} 4^p \cdot y_{pk} + \sum_{k \in K} \sum_{p \in A} \sum_{q \in C_p^A} 2^{(p+q+1)} \cdot z_{pqk} \tag{30}$$

$$z_{pqk} \geq y_{pk} + y_{qk} - 1 \qquad \forall k \in K, p \in A \text{ and } q \in C_p^A \tag{31}$$

$$\sum_{p \in A} 2^p \cdot y_{pk} \geq \delta_k \qquad \forall k \in K \tag{32}$$

$$y_{pk} \in \{0,1\} \qquad \forall k \in K \text{ and } p \in A \tag{33}$$

$$z_{pqk} \in \{0,1\} \qquad \forall k \in K, p \in A \text{ and } q \in C_p^A \tag{34}$$

Constraint set (31) in conjunction with objective function (30) ensures that the binary variable z_{pqk} takes value one when both variables y_{pk} and y_{qk} take value one, and zero, otherwise. Constraint set (32) connects binary variables y_{pk} to integer variable δ_k. Constraint sets (33) and (34) assure that decision variables y_{pk} and z_{pqk} are binary.

This method requires $\bar{m} \cdot (0.5\lfloor a \rfloor^2 + 1.5\lfloor a \rfloor + 1)$ binary variables and $\bar{m} \cdot (0.5\lfloor a \rfloor^2 + 0.5\lfloor a \rfloor + 1)$ constraints. Due to this linearization approach, the number of non-zero elements that are included into the objective function and constraints of the model is $\bar{m} \cdot (2(\lfloor a \rfloor + 1)^2 + 1)$.

The 0-1 representation method requires introduction of two types of binary variables. Another way of linearizing objective function (25) is to represent the idle time variables as 0-1-2 integer variables. Let $b = \log_3^{\bar{c}-t_{min}}$. The 0-1-2 representation of integer variable δ_k in terms of integer variables y'_{pk} is $\delta_k = \sum_{k \in B} 3^p \cdot y'_{pk}$, in which $B = \{0, 1, 2, ..., \lfloor b \rfloor\}$. Considering property (24), quadratic term δ_k^2 is represented as

$$\delta_k^2 = \sum_{p \in B} 9^p \cdot (y'_{pk})^2 + 2 \sum_{p \in B} \sum_{q \in C_p^B \setminus \{p\}} 3^{(p+q)} \cdot y'_{pk} \cdot y'_{qk}$$

in which $C_p^B = \{q \in B \mid q \geq p\}$. To linearize quadratic term $y'_{pk} \cdot y'_{qk}$, integer variable z'_{pqk} is used. It is not hard to see that the following inequality can be considered for extracting the product of two integer variables y'_{pk} and y'_{qk}.

$$2 \cdot z'_{pqk} \geq 3 \cdot (y'_{pk} + y'_{qk}) - |y'_{pk} - y'_{qk}| - 4 \qquad \forall k \in K, p \in B \text{ and } q \in C_p^B \tag{35}$$

To linearize inequalities (35), auxiliary binary variable z''_{pqk} is introduced. After linearization, minimization of the smoothness index can be executed by the use of objective function (36) and addition of constraint sets (37) through (42) into the model.

$$\min\ (u_{SX}^2 + 1) \cdot \sum_{k \in K} \delta_k + \sum_{k \in K} \sum_{p \in B} 9^p \cdot z'_{ppk} + 2 \sum_{k \in K} \sum_{p \in B} \sum_{q \in C_p^B \setminus \{p\}} 3^{(p+q)} \cdot z'_{pqk} \tag{36}$$

$$y'_{pk} + 2 \cdot y'_{qk} - 2 \cdot z''_{pqk} \leq z'_{pqk} + 2 \qquad \forall k \in K, p \in B \text{ and } q \in C_p^B \tag{37}$$

$$2 \cdot y'_{pk} + y'_{qk} + 2 \cdot z''_{pqk} \leq z'_{pqk} + 4 \qquad \forall k \in K, p \in B \text{ and } q \in C_p^B \tag{38}$$

$$\sum_{p \in B} 3^p \cdot y'_{pk} \geq \delta_k \qquad \forall k \in K \tag{39}$$

$$y'_{pk} \in \{0,1,2\} \qquad \forall k \in K \text{ and } p \in B \tag{40}$$

$$z'_{pqk} \in \{0,1,2,3,4\} \qquad \forall k \in K, p \in B \text{ and } q \in C_p^B \tag{41}$$

$$z''_{pqk} \in \{0,1\} \qquad \forall k \in K, p \in B \text{ and } q \in C_p^B \tag{42}$$

Constraint sets (37) and (38) in conjunction with objective function (36) ensure that the binary variable z'_{pqk} takes the product of two integer variables y'_{pk} and y'_{qk}. These constraints are the linearized form of inequalities (35). Constraint set (39) connects integer variables y'_{pk} to integer variables δ_k. Constraint sets (40), (41) and (42) define the new decision variables of the model.

In total, the 0-1-2 representation method requires introduction of $\bar{m}\cdot(\lfloor b\rfloor^2+4\lfloor b\rfloor+3)$ integer variables and $\bar{m}\cdot(\lfloor b\rfloor^2+3\lfloor b\rfloor+3)$ constraints. Due to this linearization, the number of non-zero elements that are included into the objective function and constraints of the model is $\bar{m}\cdot(4.5\lfloor b\rfloor^2+12.5\lfloor b\rfloor+9)$. Note that in constraint sets (37) and (38) when $p=q$, only one coefficient ($=3$) for variable y'_{pk} is considered. Table 3 numerically compares the three methods in terms of the number of variables (NV), the number of constraints (NC), and the number of non-zero elements (NNZ). Since $b=a\cdot\log_3^2$, the problem size is only described by factor a.

Table 3: The comparison of the three linearization methods

a	$NV(\times\bar{m})$			$NC(\times\bar{m})$			$NNZ(\times\bar{m})$		
	TI	0-1	0-1-2	TI	0-1	0-1-2	TI	0-1	0-1-2
5	32	21	24	2	16	21	97	73	87
10	1024	66	63	2	56	57	3073	243	246
15	32768	136	120	2	121	111	98305	513	486

The data in the table show that the time index representation method include many variables and non-zero elements into the model in comparison to the other two methods. However, it requires a few numbers of constraints, irrespective of the value of a. The difference between the 0-1 and the 0-1-2 methods is small enough to say that they have similar size complexity. However, later in section 5, the experimental results indicate that the 0-1 method has very small computational complexity in comparison to the 0-1-2 method.

4 Formulation improvement

This section discusses how the proposed formulation can be improved. There are different techniques to enhance a mathematical programming model. Some techniques such as preprocessing and cut generation are widely implemented in the integer programming (IP) solvers to make the original formulation tighter. However, sometimes a mathematical programming formulation can be strengthened by defining special preprocessing and valid inequalities for the formulation. These are usually inspired by the specific knowledge of the problem. This paper also provides two techniques to improve the presented formulation. The first technique is based on generating valid inequalities based on the precedence relations, and the second is a new approach to speed up solving the linear programming (LP) relaxations of the integer program.

4.1 Precedence-oriented inequalities

In section 2, we show that similar to the SALBP-1 and SALBP-2, a preprocessing based on the concept of earliest and latest stations can be applied to the SALBP-E. The performance of this preprocessing depends primarily on the bounds provided for the cycle time and the number of stations. Usually these bounds are weak, and therefore a few numbers of variables are eliminated in the preprocessing phase.

In the following, we present two new sets of valid inequalities based on the precedence relations. These inequalities impose the concept of earliest and latest stations on the formulation dynamically, i.e. with the bounds provided by the formulation itself. The following set of inequalities ensures that task i cannot be assigned to some earlier stations.

$$(t_i + t_i^P) \cdot x_{ik} \leq k \cdot c \qquad \forall i \in V \text{ and } k \in FS_i \tag{43}$$

To show the validity of these inequalities, suppose that c is given. By dividing both sides of inequalities (43) by c, the coefficient of variable x_{ik} is a lower bound on the station index to which task i can be assigned (see the definition of the earliest station in Table 1). So, if task i and its predecessors require more than k stations, then it cannot be assigned to station k ($x_{ik} = 0$). Since c is a decision variable, we write inequalities (43) in such a form to be able to impose this concept on the formulation by linear constraints. In fact, inequalities (43) give a lower bound on the cycle time based on the task assignments to the stations. These inequalities can be strengthened by considering the idle times as follows.

$$(t_i + t_i^P) \cdot x_{ik} + \sum_{k'=1}^{k} \delta_{k'} \leq k \cdot c \qquad \forall i \in V \text{ and } k \in FS_i \tag{44}$$

In other words, if the execution time of task i and its predecessors plus the idle times of stations 1 through k require more than k stations, variable x_{ik} cannot take value one.

Another set of inequalities can be developed to restrict the set of tasks assignable to the last stations. These inequalities can be expressed in terms of the required line capacity as follows.

$$(t_i + t_i^F) \cdot x_{ik} + (k-1) \cdot c \leq T \qquad \forall i \in V \text{ and } k \in (FS_i \cap KD)\backslash\{1\} \tag{45}$$

$$(t_i + t_i^F) \cdot x_{ik} + (k-1) \cdot c \leq T + ((k-1) \cdot \bar{c} - t_{sum}) \cdot (1 - u_k) \qquad \forall i \in V \text{ and } k \in (FS_i \cap KD) \tag{46}$$

Suppose that combination (m, c) is given. According to inequalities (45) and (46), task i cannot be assigned to station k if this assignment violates the given capacity. To better understand these inequalities, consider inequalities (45). Replace T with $m \cdot c$, and divide both sides of the inequalities by c. The result is as follows:

$$k \leq m + 1 - (\frac{t_i + t_i^F}{c}) \cdot x_{ik}$$

As is seen, inequalities (45) specify an upper bound on the station index to which task i can be assigned (see the definition of the latest station in Table 1). So, if task i is assigned to station k, then $k-1$ previous stations plus the number of required stations for task i and its successors cannot violate the total number of stations, m. If no task is assigned to station k, inequalities (46) remain inactive. Otherwise, they are similar to inequalities (45). Note that similar to what we have done for strengthening inequalities (43), inequalities (45) and (46) can be strengthened by considering the idle times as follows:

$$(t_i + t_i^F) \cdot x_{ik} + (k-1) \cdot c + \sum_{k'=k}^{\bar{m}} \delta_{k'} \leq T \qquad \forall i \in V \text{ and } k \in (FS_i \cap KD)\backslash\{1\} \tag{47}$$

$$(t_i + t_i^F) \cdot x_{ik} + (k-1) \cdot c + \sum_{k'=k}^{\bar{m}} \delta_{k'} \leq T + ((k-1) \cdot \bar{c} - t_{sum}) \cdot (1 - u_k) \qquad \forall i \in V \text{ and } k \in (FS_i \cap KD) \tag{48}$$

Such strengthening of these inequalities is more understood when we consider their linearized form as follows. Observe that T is replaced with $\sum_{k\in K}\delta_k + t_{sum}$.

$$(t_i + t_i^F)\cdot x_{ik} + (k-1)\cdot c \leq \sum_{k'=1}^{k-1}\delta_{k'} + t_{sum} \qquad \forall i \in V \text{ and } k \in (FS_i \cap KD)\backslash\{1\} \tag{49}$$

$$(t_i + t_i^F)\cdot x_{ik} + (k-1)\cdot c \leq \sum_{k'=1}^{k-1}\delta_{k'} + t_{sum} + ((k-1)\cdot \bar{c} - t_{sum})\cdot(1-u_k) \quad \forall i \in V \text{ and } k \in (FS_i \cap KD) \tag{50}$$

In Section 5, we show that adding inequalities (44), (49), and (50) into the formulation significantly reduces the computational complexity of the model.

4.2 Size reduction by the use of auxiliary variables

The quality of a linear programming problem depends on the size of the coefficient matrix, which is a function of the number of variables and constraints, as well as the number of non-zero elements (see Chen et al. [6]). A high number of non-zeros in the coefficient matrix can cause numerical difficulties to solve the LP relaxations. In this section, we introduce a new approach to decrease problem size by reducing the number of non-zero elements in the coefficient matrix. We show that defining *auxiliary variables* can be very effective in reducing the problem size.

Auxiliary variables are commonly used in integer programming as well as constraint programming methods for different reasons. For example, they are used in formulating non-convex decisions, guiding branching scheme, or hybrid modeling. We refer the interested reader to Smith [25], Atamtürk and Savelsbergh [2], and Hooker [14] for some applications of the auxiliary variables. In the following, it is shown that how auxiliary variables can be used in modeling a better formulation.

Let $|R|$ and $|FS_i|$ be the cardinality of sets R and FS_i, respectively. Observe that $|R|$ and $|FS_i|$ are bounded by $O(n^2)$ and $O(n)$, respectively. In FE, precedence constraints (7) add a great number of non-zero elements into the coefficient matrix. In fact, they add $N_1 = \sum_{(i,j)\in R}(|FS_i| + |FS_j|)$ non-zero elements to the coefficient matrix of the formulation, which is bounded by $O(n^3)$. It is not hard to see that by eliminating these constraints, the number of non-zero elements of FE is decreased to $O(n^2)$.

To alleviate the numerical difficulties associated with precedence constraints (7), we define continuous auxiliary variable α_i as follows:

$$\alpha_i = \sum_{k\in FS_i} k\cdot x_{ik} \qquad \forall i \in V \tag{51}$$

where α_i indicates the station number to which task i is assigned. Using these auxiliary variables we can write precedence constraints as follows:

$$\alpha_i \leq \alpha_j \qquad \forall (i,j)\in R \tag{52}$$

Therefore, constraint set (7) is replaced with constraint sets (51) and (52). Note that terms $\sum_{k\in FS_i} k\cdot x_{ik}$ still remain in the formulation, but in much smaller number of constraints. In constraint set (7) they are considered for each $(i,j)\in R$, but in constraint sets (51) they are considered for each $i \in V$. In fact, the number of non-zero elements that constraint sets (51) and (52) add to the formulation is equal to $N_2 = n + 2|R| + \sum_{i\in V}|FS_i|$, which is bounded by $O(n^2)$.

Also, it is important to point out that adding valid inequalities (44), (49), and (50) to the formulation makes it stronger, albeit with a sharp increase in the number of non-zero elements. Thus, to avoid such inefficiency, we define continuous auxiliary variable β_k as follows.

$$\beta_k = \sum_{k'=1}^{k} \delta_{k'} \qquad \forall k \in K \tag{53}$$

By these auxiliary variables, the modified inequalities are expressed as follows:

$$(t_i + t_i^P) \cdot x_{ik} + \beta_k \leq k \cdot c \qquad \forall i \in V \text{ and } k \in FS_i \tag{54}$$

$$(t_i + t_i^F) \cdot x_{ik} + (k-1) \cdot c \leq \beta_{k-1} + t_{sum} \qquad \forall i \in V \text{ and } k \in (FS_i \cap KD) \backslash \{1\} \tag{55}$$

$$(t_i + t_i^F) \cdot x_{ik} + (k-1) \cdot c \leq \beta_{k-1} + t_{sum} + ((k-1) \cdot \bar{c} - t_{sum}) \cdot (1 - u_k) \qquad \forall i \in V \text{ and } k \in (FS_i \cap KD) \tag{56}$$

Note that defining the auxiliary variables adds $n + \bar{m}$ variables and $n + \bar{m}$ constraints to the improved formulation, but it reduces the number of non-zero elements to $O(n^2)$. The effectiveness of this technique is investigated in Section 5. Such application of the auxiliary variables is also applied to an order acceptance and scheduling problem by Esmaeilbeigi et al. [10].

5 Computational Results

To evaluate the performance of the proposed formulation and improvements, we conducted a computational study on the benchmark data sets. We used CPLEX 12.4 to solve the model. The time limit was set to 1800 seconds. All the experiments were run on a computer with a single-core 2.5 GHz Intel processor and 4.0 GB RAM.

The well-known instances of the SALBP, including the SALBP-E instances, can be found in the website `http://alb.mansci.de`. We consider two classes of instances of the problem, denoted by C1 and C2. Class C1 includes small and medium instances with up to 50 tasks, and Class C2 includes larger instances from 50 to 100 tasks. In total, C1 and C2 include 62 and 103 instances of the SALBP-E, respectively. In each class, a set of precedence graphs, and for each precedence graph a number of instances exists. Each precedence graph is specified with a “Name”, and each instance is specified with a lower bound and an upper bound on the number of stations. The bounds on the cycle time are also calculated through Proposition 1.

To show the effectiveness of the proposed improvements, they are separately added to the formulation. Consequently, four different levels of the formulation are considered as follows.

- FE.1: The original formulation of FE;
- FE.2: FE.1 when non-negativity constraints (20) are changed to integrality constraints.
- FE.3: FE.2 with valid inequalities (44), (49), and (50).
- FE.4: FE.2 when constraints (51) and (52) are used instead of constraints (7) , and constraints (54), (55), and (56) are added.

Also, to compare the three linearization methods of the objective function (25), three levels FE.5, FE.6, and FE.7 are defined as follows.

- FE.5: FE.4 in which objective function (26) is used, and constraint sets (27), (28), and (29) are added.

- FE.6: FE.4 in which objective function (30) is used, and constraint sets (31), (32), (33), and (34) are added.
- FE.7: FE.4 in which objective function (36) is used, and constraint sets (37), (38), (39), (40), (41), and (42) are added.

Moreover, to examine the impact of the two other secondary objectives on the formulation, we consider two levels of formulation as follows.

- FE.8: FE.4 in which objective function (21) is used.
- FE.9: FE.4 in which objective function (22) is used.

In order to measure the performance of each formulation, the average computational time ($Avg.Time$) and the average number of searched nodes ($Avg.Node$) are considered. Where suitable, the average optimality gap ($Avg.Gap$) and the number of instances which are solved to optimality are also reported. We first examine the effect of the proposed inequalities on the formulation.

5.1 Performance of valid inequalities

In order to show the effectiveness of the proposed inequalities, and to examine the effect of the integrality consideration of the idle time variables, we solve instances of C1 by FE.1, FE.2 and FE.3. Details of the results are provided in Table 4. As was mentioned, for each precedence graph a number of instances exists. So, for each precedence graph in the table (each row), the averages are reported. In the row AVG, the weighted averages (affected by the number of instances for each precedence graph) are shown. In other words, in the row AVG the average amounts for all 62 instances of the class are reported.

We first compare FE.1 with FE.2 to evaluate performance of the general cutting planes generated by CPLEX due to integrality of the idle time variables. Note that we make CPLEX to generate such cuts by changing the non-negativity constraints of the idle time variables to integrality constraints. The data in the table indicates that these cutting planes result in a significant improvement in the runtime. It can be seen that FE.1 cannot solve all problem instances within the specified time limit (two instances of "Sawyer" and two instances of "Kilbridge" are not solved by FE.1). Evidently, in terms of the solution time, FE.2 performs far better than FE.1 because the number of searched nodes in FE.2 is reduced considerably. For example, in the specified time limit, CPLEX explores 861801 nodes to solve instances of "Kilbridge" by FE.1, but it explores 4091 nodes to solve them by FE.2 (99.5% reduction in the number of searched nodes). This confirms that general cutting planes are very effective in solving the FE. These cutting planes improve the global lower bound on the total idle time, significantly. Note that general cutting planes are usually expected to be very weak (see e.g., Chen et al. [7]), but as we have shown, they give good performance for the FE.

We now compare FE.2 with FE.3 to examine effect of the precedence-oriented inequalities on FE.2. Note that both FE.2 and FE.3 solve all the instances to optimality. As is shown by Table 4, these valid inequalities reduce the number of explored nodes in all instances. For example, in FE.3 the number of searched nodes for instances of "Kilbridge" is reduced to 1025 nodes, which shows a 75% decrease in comparison to FE.2. In sum, the average solution time for 62 instances of C1 is reduced by more than 60%. These experimental results confirm that the proposed valid inequalities make the original formulation stronger.

5.2 Role of the auxiliary variables

To show that how auxiliary variables can be effective in solving instances of the problem, we compare FE.3 with FE.4 on C2. Details for these instances are provided in Table 5. For the better comparison, the numbers of instances which are solved to optimality are also shown.

Table 4: Comparing FE.1, FE.2, and FE.3 on C1.

Name	FE.1			FE.2		FE.3	
	Avg.Time *(s)*	*Avg.Gap* *(%)*	*Avg.Node* *NO*	*Avg.Time* *(s)*	*Avg.Node* *NO*	*Avg.Time* *(s)*	*Avg.Node* *NO*
"Mertens"	0.1	0	103	0.1	68	0.1	68
"Bowman"	0.1	0	13	0.1	14	0.1	5
"Jaeschke"	0.1	0	100	0.1	69	0.2	23
"Jackson"	0.2	0	460	0.2	270	0.1	32
"Mansoor"	0.1	0	107	0.1	89	0.1	15
"Mitchell"	0.4	0	381	0.8	774	0.4	100
"Roszieg"	3.5	0	3850	3.6	4470	0.8	485
"Heskiaoff"	289.2	0	696182	27.7	55220	17.4	20775
"Buxey"	207.7	0	169244	45.9	32853	7.8	4883
"Sawyer"	714.6	25.0	528396	44.7	44816	34.0	31361
"Lutz1"	57.6	0	27324	46.1	23975	8.8	5550
"Gunther"	301.8	0	132851	67.6	35008	19.3	8185
"Kilbridge"	647.9	33.3	861801	5.0	4091	2.7	1025
AVG	**251.1**	**6.5**	**258156**	**27.6**	**22430**	**10.7**	**8344**

Table 5: Comparing FE.3 with FE.4 on C2.

Name	FE.3				FE.4			
	Avg.Time *(s)*	*Avg.Gap* *(%)*	*Avg.Node* *NO*	*Optimal* *NO*	*Avg.Time* *(s)*	*Avg.Gap* *(%)*	*Avg.Node* *NO*	Optimal *NO*
"Hahn"	1.5	0	531	4	1.5	0	379	4
"Warnecke"	1380.0	69.1	79959	4	1157.8	55.6	200129	6
"Tonge"	1781.1	84.7	553871	1	1650.6	83.9	533367	1
"Wee-Mag"	1557.7	79.3	135882	3	1444.3	70.8	111403	5
"Arcus1"	1679.7	53.8	275812	1	1663.9	53.3	447361	1
"Lutz2"	1488.2	66.1	39710	4	1438.9	61.3	140658	6
"Lutz3"	833.5	14.3	81564	12	386.1	0	67547	14
AVG	**1389.1**	**59.5**	**167073**	**28.2%**	**1241.6**	**52.6**	**217475**	**35.9%**

According to Table 5, FE.3 is able to solve 28.2% of instances of C2 to optimality, but FE.4 solves 35.9%. Except for instances of "Hahn" which are solved to optimality by both FE.3 and FE.4, in the other cases, FE.4 shows better performance in terms of the solution time and the optimality gap. We now compare the number of nodes which are explored by CPLEX for solving FE.3 and FE.4. It can be seen that CPLEX explores 167073 nodes within 1389.1 seconds to solve FE.3. However, it explores 217475 nodes within 1241.6 seconds to solve FE.4. These data say that CPLEX explores 120 nodes per second to solve FE.3, but it explores 175 nodes per second to solve FE.4. Thus, due to including the auxiliary variables, the LP relaxations for FE.4 are easier to be solved. Such advantage of using the auxiliary variables can improve both the global lower bound and the global upper bound in the B&B tree. For example, consider instance "Lutz3" with $(\underline{m}, \bar{m}) = (11, 23)$ and the optimal objective value of 6. According to our computational experiments, CPLEX explores 41740 nodes within 1800 seconds to solve this instance using FE.3. However, the instance cannot be solved to optimality and CPLEX achieves interval $[0, 24]$ for the optimal solution. When FE.4 is used, CPLEX explores 125516 nodes, and achieves the optimal solution within 972 seconds. In Table 6, we report details for 29 instances of

C2 which are solved to optimality by both FE.3 and FE.4.

Table 6: Results of instances in C2 which are solved to optimality by both FE.3 and FE.4.

Name	FE.3		FE.4	
	Avg.Time (s)	*Avg.Node NO*	*Avg.Time (s)*	*Avg.Node NO*
"Hahn"	1.5	531	1.5	379
"Warnecke"	224.8	74061	77.4	55561
"Tonge"	1572.5	467903	5.8	2209
"Wee-Mag"	262.0	76776	605.5	83995
"Arcus1"	236.3	19362	29.9	2311
"Lutz2"	396.8	25293	256.6	70779
"Lutz3"	672.4	83920	365.9	68094
AVG	**453.6**	**73247**	**261.6**	**54500**

It can be inferred from Table 6 that on average the solution time is improved by 42.3% for these instances. In addition, in the most cases the use of the auxiliary variables has reduced the number of searched nodes as well. We also solved instances of C1 using auxiliary variables. Even in these instances the average solution time and the average number of searched nodes are reduced, i.e. to 8.6 seconds and 7322 nodes (see Table 8). It is evident from these results that the introduction of auxiliary variables is effective in solving the FE.

5.3 Comparison of the three linearization methods

In Section 3, we see that the 0-1 and the 0-1-2 linearization methods are similar in terms of the size. Moreover, it is shown that the time index representation method adds a lot of variables and non-zero elements. Table 7 compares the three methods on C1, computationally. Although, FE.6 and FE.7 are similar in size, FE.7 cannot solve all instances to optimally, and its solution time is about 14 times larger than FE.6. This is because CPLEX can solve the binary programs more efficient than the integer programs.

As is seen from the table, FE.5 and FE.6 are similar in terms of the computational time, but they mainly differ in the number of searched nodes. In fact, CPLEX explores 238 and 549 nodes per second to solve FE.5 and FE.6, respectively. This is because FE.5 adds a great number of non-zero elements, which increases the solution time of the LP-relaxations. The results indicate that the 0-1 representation method is better because it requires very low memory in comparison to the time index representation method.

5.4 Effects of the secondary objectives

In order to examine impact of the secondary objectives on the formulation, we used C1 and solved FE.4, FE.8, and FE.9. The results are summarized in Table 8. It can be seen that in comparison to FE.4, solution time for FE.6, FE.8, and FE.9 is increased by a factor of 4.8, 2.6, and 1.3, respectively. While minimizing the cycle time as a secondary objective makes the formulation hard to solve, minimizing the number of stations is rather easier. Likely the reason is that in the optimal solution for these instances, the optimal number of stations is very close to the given lower bound on the number of stations. For example, in the results of solving FE.8, in 55% of instances, the optimal number of stations is equal to the lower bounds. Thus, considering minimization of the cycle time as a secondary objective in these instances is in conflict with the original objective of the problem, and makes the problem harder to

Table 7: Computational comparison of the three linearization methods on C1.

Name	FE.7			FE.6		FE.5	
	Avg.Time (*s*)	*Avg.Gap* ($\times 10^{-4}$%)	*Avg.Node* *NO*	*Avg.Time* (*s*)	*Avg.Node* *NO*	*Avg.Time* (*s*)	*Avg.Node* *NO*
"Mertens"	0.2	0	369	0.2	67	0.1	62
"Bowman"	0.2	0	85	0.3	13	0.1	12
"Jaeschke"	0.6	0	1510	0.3	82	0.2	58
"Jackson"	0.3	0	289	0.3	55	0.2	58
"Mansoor"	0.3	0	196	0.2	13	0.3	26
"Mitchell"	1.3	0	2398	0.5	155	0.6	111
"Roszieg"	193.0	0	203252	1.3	458	1.4	598
"Heskiaoff"	1029.0	0.7	961129	49.9	31735	58.6	30995
"Buxey"	388.5	2.9	404251	15.0	12488	16.7	9871
"Sawyer"	783.5	4.6	806013	29.6	15726	26.4	19072
"Lutz1"	1801.2	0.1	602582	46.4	9336	222.3	5521
"Gunther"	923.6	8.3	544092	196.8	114074	34.1	12707
"Kilbridge"	302.2	0.5	598841	4.9	2592	4.6	1325
AVG	579.9	2.1	442528	40.9	22434	37.5	8933

solve. FE.6 is much harder to solve because the size of the model is increased due to the linearization scheme. Evidently, the increase in the solution time and the number of searched nodes are penalty for considering the secondary objectives.

Table 8: Effect of considering secondary objectives on instances of C1.

Name	FE.4		FE.8		FE.9	
	Avg.Time (*s*)	*Avg.Node* *NO*	*Avg.Time* (*s*)	*Avg.Node* *NO*	*Avg.Time* (*s*)	*Avg.Node* *NO*
"Mertens"	0.1	62	0.3	100	0.1	69
"Bowman"	0.2	17	0.1	5	0.1	29
"Jaeschke"	0.2	34	0.3	66	0.3	47
"Jackson"	0.2	105	0.3	229	0.3	58
"Mansoor"	0.1	20	0.2	0	0.2	39
"Mitchell"	0.5	21	1.0	479	0.6	229
"Roszieg"	0.8	22	1.7	1153	1.2	698
"Heskiaoff"	16.6	23	35.7	22500	36.3	29563
"Buxey"	3.9	24	22.0	9891	5.2	3976
"Sawyer"	20.7	25	86.1	33773	25.1	23256
"Lutz1"	7.5	26	8.1	4189	7.0	4711
"Gunther"	18.8	27	25.1	11453	20.9	9656
"Kilbridge"	5.8	28	8.0	3176	3.8	1602
AVG	**8.6**	29	**22.1**	**9967**	**11.2**	**8237**

6 Conclusion

In this study, we developed a mixed integer linear programming formulation for the SALBP-E. In comparison to the previous formulations of the SALBP-1 and SALBP-2, the number of additional variables and the number of additional constraints required for this formulation is linearly bounded by the number of tasks. We showed that considering general cutting planes and precedence-oriented valid inequalities are very effective in solving the presented formulation. In addition, including appropriate auxiliary variables into the formulation is shown to be helpful in reducing the solution time. These auxiliary variables decrease the number of non-zero elements in the coefficient matrix of the formulation considerably. The improved formulation is then capable of solving instances of the problem with up to 50 tasks in less than 10 seconds on average. The effect of considering different secondary objectives is also examined on the improved formulation. In the case of workload smoothing, three different linearization methods are employed and compared for minimizing the smoothness index.

The future study can be devoted to improving the current formulation. For example, it is possible that new sets of valid inequalities are developed for the problem. Also, one may reformulate the problem and provide a better formulation by new sets of decision variables. Valid inequalities might have a better effect if the problem is modeled in another way. Moreover, in this paper, each secondary objective is considered as part of a single objective function. The secondary objectives can also be optimized lexicographically, i.e., by procedures that solve a sequence of single-objective optimization problems. Both procedures require more research and mathematical analysis.

References

[1] Amen, M. (2006). Cost-oriented assembly line balancing: Model formulations, solution difficulty, upper and lower bounds. *European Journal of Operational Research*, 168(3):747 – 770. Balancing Assembly and Transfer lines.

[2] Atamtürk, A. and Savelsbergh, M. (2005). Integer-programming software systems. *Annals of Operations Research*, 140(1):67–124.

[3] Battaa, O. and Dolgui, A. (2013). A taxonomy of line balancing problems and their solutionapproaches. *International Journal of Production Economics*, 142(2):259 – 277. Anticipation of risks impacts and industrial performance evaluation in distributed organizations life cycles.

[4] Baybars, l. (1986). A survey of exact algorithms for the simple assembly line balancing problem. *Management Science*, 32(8):909–932.

[5] Bowman, E. H. (1960). Assembly-line balancing by linear programming. *Operations Research*, 8(3):385–389.

[6] Chen, D.-S., Batson, R. G., and Dang, Y. (2009a). *Better Formulation by Preprocessing*, pages 79–104. John Wiley & Sons, Inc.

[7] Chen, D.-S., Batson, R. G., and Dang, Y. (2009b). *Branch-and-Cut Approach*, pages 305–333. John Wiley & Sons, Inc.

[8] Emde, S., Boysen, N., and Scholl, A. (2010). Balancing mixed-model assembly lines: a computational evaluation of objectives to smoothen workload. *International Journal of Production Research*, 48(11):3173–3191.

[9] Erel, E. and Sarin, S. C. (1998). A survey of the assembly line balancing procedures. *Production Planning & Control*, 9(5):414–434.

[10] Esmaeilbeigi, R., Charkhgard, P., and Charkhgard, H. (preprint 2014). Order acceptance and scheduling problem in two-machine flow shops: New mixed integer programming formulations. `http://www.optimization-online.org/DB_FILE/2014/07/4468.pdf`.

[11] Eswaramoorthi, M., Kathiresan, G., Jayasudhan, T., Prasad, P., and Mohanram, P. (2012). Flow index based line balancing: a tool to improve the leanness of assembly line design. *International Journal of Production Research*, 50(12):3345–3358.

[12] Ghosh, S. and Gagnon, R. J. (1989). A comprehensive literature review and analysis of the design, balancing and scheduling of assembly systems. *International Journal of Production Research*, 27(4):637–670.

[13] Hackman, S. T., Magazine, M. J., and Wee, T. S. (1989). Fast, effective algorithms for simple assembly line balancing problems. *Operations Research*, 37(6):916–924.

[14] Hooker, J. (2011). Hybrid modeling. In van Hentenryck, P. and Milano, M., editors, *Hybrid Optimization*, volume 45 of *Springer Optimization and Its Applications*, pages 11–62. Springer New York.

[15] Mozdgir, A., Mahdavi, I., Badeleh, I. S., and Solimanpur, M. (2013). Using the taguchi method to optimize the differential evolution algorithm parameters for minimizing the workload smoothness index in simple assembly line balancing. *Mathematical and Computer Modelling*, 57(12):137 – 151. Mathematical and Computer Modelling in Power Control and Optimization.

[16] Pastor, R. and Ferrer, L. (2009). An improved mathematical program to solve the simple assembly line balancing problem. *International Journal of Production Research*, 47(11):2943–2959.

[17] Patterson, J. H. and Albracht, J. J. (1975). Technical noteassembly-line balancing: Zero-one programming with fibonacci search. *Operations Research*, 23(1):166–172.

[18] Rachamadugu, R. and Talbot, B. (1991). Improving the equality of workload assignments in assembly lines. *International Journal of Production Research*, 29(3):619–633.

[19] Rekiek, B., Dolgui, A., Delchambre, A., and Bratcu, A. (2002). State of art of optimization methods for assembly line design. *Annual Reviews in Control*, 26(2):163 – 174.

[20] Saif, U., Guan, Z., Wang, B., Mirza, J., and Huang, S. (2014). A survey on assembly lines and its types. *Frontiers of Mechanical Engineering*, 9(2):95.

[21] Scholl, A. (1999). *Balancing and Sequencing of Assembly Lines*. Contributions to Management Science. Physica-Verlag HD.

[22] 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*, 168(3):666 – 693. Balancing Assembly and Transfer lines.

[23] Scholl, A., Boysen, N., and Fliedner, M. (2013). The assembly line balancing and scheduling problem with sequence-dependent setup times: problem extension, model formulation and efficient heuristics. *OR Spectrum*, 35(1):291–320.

[24] Sivasankaran, P. and Shahabudeen, P. (2014). Literature review of assembly line balancing problems. *The International Journal of Advanced Manufacturing Technology*, 73(9-12):1665–1694.

[25] Smith, B. M. (2005). Modelling for constraint programming. *Lecture Notes for the First International Summer School on Constraint Programming.*

[26] Wei, N.-C. and Chao, I.-M. (2011). A solution procedure for type e simple assembly line balancing problem. *Computers & Industrial Engineering*, 61(3):824 – 830.

[27] White, W. W. (1961). Letter to the editorcomments on a paper by bowman. *Operations Research*, 9(2):274–276.

[28] Zacharia, P. and Nearchou, A. C. (2013). A meta-heuristic algorithm for the fuzzy assembly line balancing type-e problem. *Computers & Operations Research*, 40(12):3033 – 3044.