OR Spectrum (2016) 38:493–518
DOI 10.1007/s00291-016-0433-3

REGULAR ARTICLE

New formulations for the setup assembly line balancing and scheduling problem

Rasul Esmaeilbeigi[1] · **Bahman Naderi**[1] · **Parisa Charkhgard**[2]

Received: 2 November 2014 / Accepted: 20 January 2016 / Published online: 8 February 2016

Abstract We present three new formulations for the setup assembly line balancing and scheduling problem (SUALBSP). Unlike the simple assembly line balancing problem, sequence-dependent setup times are considered between the tasks in the SUALBSP. These setup times may significantly influence the station times. Thus, there is a need for scheduling the list of tasks within each station so as to optimize the overall performance of the assembly line. In this study, we first scrutinize the previous formulation of the problem, which is a *station-based* model. Then, three new formulations are developed by the use of new sets of decision variables. In one of these formulations, the *schedule-based* formulation, SUALBSP is completely formulated as a scheduling problem. That is, no decision variable in the model directly denotes a station. All the proposed formulations will be improved by the use of several enhancement techniques such as preprocessing and valid inequalities. These improved formulations can be applied to establishing lower bounds on the problem. To assess the performance of new formulations, results of an extensive computational study on the benchmark data sets are also reported.

Keywords Setup assembly line balancing and scheduling problem · Mathematical formulation

✉ Bahman Naderi
bahman.naderi@aut.ac.ir

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

[2] School of Mathematical and Physical Sciences, The University of Newcastle, Newcastle, Australia

1 Introduction

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 1998). Usually, in the ALBPs the objective is to minimize the number of required stations for a given cycle time (or for a given production rate), which is called type-1. This objective is the most frequently employed in literature (Sewell and Jacobson 2012; Battaïa and Dolgui 2013; Dolgui et al. 2014; Kucukkoc and Zhang 2015; Pape 2015). The problem is called type-2 if the number of stations is given a priori, and the objective is to maximize the production rate through minimizing the cycle time. In type-E, both of the cycle time and total number of stations are decision variables, and the objective is to maximize line efficiency.

The basic version of the ALBP is referred to as the simple assembly line balancing problem (SALBP). In the SALBP, a single product (also called single-model) is manufactured on a serial line. Moreover, it is assumed that task times are deterministic and integer values. The total execution time of tasks assigned to a station constitutes the station time which must not exceed the cycle time. Bowman (1960) develops two different integer linear programming formulations for SALBP type-1 (SALBP-1). Later, White (1961) and Patterson and Albracht (1975) present more efficient formulations for the SALBP-1. Considering White's formulation, Baybars (1986) presents a binary integer linear programming formulation for SALBP type-2 (SALBP-2). To improve mathematical formulations of both SALBP-1 and SALBP-2, Pastor and Ferrer (2009) introduce new sets of valid inequalities. A mixed integer linear programming formulation for SALBP type-E (SALBP-E) is developed by Esmaeilbeigi et al. (2015b). They also present new sets of valid inequalities to strengthen this formulation.

Andrés et al. (2008) extend the SALBP-1 to the assembly line balancing and scheduling with sequence-dependent setup times. In this problem, unlike the classical SALBP, the sequence of tasks within each station may influence the station times. A station time is now calculated as the total execution time of tasks assigned to the station plus their sequence-dependent setup times. The setup time consideration in Andrés et al. (2008) is along the following reasoning: "whenever a task is assigned next to another at the same workstation, a setup time must be added to compute the global workstation time". The scheduling problem inside each station can be considered as a traveling salesman problem (TSP) with some forbidden paths (because of the precedence relations) in which the setup times are the distances between the cities. They formulate the problem as a binary linear program with $O(n^3)$ variables and $O(n^4)$ constraints and solve very small instances of the problem to optimality. To evaluate efficiency of their proposed heuristics, they use the lower bounds provided by the formulation after a certain computational time. Martino and Pastor (2010) propose additional heuristics based on several priority rule-based procedures to solve this problem.

Scholl et al. (2013) extend the work of Andrés et al. (2008) to the setup assembly line balancing and scheduling problem type-1 (SUALBSP-1), where forward and backward setup times are distinguished. The forward setup time occurs when a task directly follows another task in the same station and at the same cycle, i.e., on the same workpiece. The backward setup time, on the other hand, exists between the last task in a cycle p and the first task in the following cycle $p+1$ at the same station. Scholl et al. (2013) also state that a (modified) triangle inequality is fulfilled in almost any case in practice, i.e., the station time is not decreased whenever an additional task is included. They present a new mixed integer linear programming formulation for the problem. The number of variables and constraints in this formulation is bounded by $O(n^2)$ that shows a great size reduction in comparison to the proposed model of Andrés et al. (2008). However, they do not report numerical results for this formulation.

The literature review indicates that SUALBSP is the most relevant problem when taking setup times into account. The former research focuses on incompatible tasks, i.e. tasks that must not be assigned to the same station for any reason (see e.g., Becker and Scholl 2009; Scholl et al. 2010). In addition, Scholl et al. (2008) study the sequence-dependent assembly line balancing problem (SDALBP) where sequence-dependent task time increments are defined. The SDALBP has its own practical motivation and assumptions. The interested reader is referred to Scholl et al. (2008) for the details on SDALBP. Evidently, the solution to the SUALBSP is the best way to deal with setup times in the assembly line balancing context.

For SUALBSP type-2 (SUALBSP-2), a hybrid genetic algorithm is proposed by Yolmeh and Kianfar (2012). To the best of our knowledge, no exact solution procedure exists in the literature to solve the SUALBSP-2. For the SUALBSP-1, the only exact solution approach is the mathematical formulation of Scholl et al. (2013). This formulation will be completed and improved in this paper. In addition to the improved formulation, two other formulations are presented for the SUALBSP-1. One of these formulations is a station-based model and the other is a schedule-based one. In the schedule-based model, the problem is completely formulated as a scheduling problem. We also show that the proposed formulations of the SUALBSP-1 can be employed to solve the SUALBSP-2. All the proposed formulations will be improved by the use of several enhancement techniques such as preprocessing and valid inequalities.

The rest of the paper is organized as follows. In Sect. 2 we review and scrutinize the previous formulation of the SUALBSP. In this section, we indicate that the formulation is incomplete and may result in incorrect solutions. Then, new sets of constraints are introduced to make the formulation complete. Section 3 provides three new formulations for the problem. To improve these formulations, some methods are developed in Sect. 4. In Sect. 5, we present results of a computational study on the benchmark data set to assess performance of the proposed formulations. Finally, the paper will be concluded in Sect. 6.

2 A review and scrutiny of the previous formulation

In the SUALBSP two types of decisions must be made simultaneously: how to assign tasks to the stations and how to schedule the list of tasks within each station. The

Table 1 The general parameters of the SUALBSP

Notation	Definition
n	The number of tasks
V	Set of tasks, i.e., $V = \{1, 2, \dots, n\}$
i, j, v	Index for the tasks
k	Index for the stations
$E(E^*)$	Set of direct (all) precedence relations
t_i	Execution time for task $i \in V$
t_{sum}	The sum of the task times, i.e., $t_{\text{sum}} = \sum_{i \in V} t_i$
$P_i(P_i^*)$	Set of direct (all) predecessors of task $i \in V$
$F_i(F_i^*)$	Set of direct (all) successors of task $i \in V$
$\bar{c}(\underline{c})$	Upper (lower) bound on the cycle time
$\bar{m}(\underline{m})$	Upper (lower) bound on the number of stations
E_i	Earliest station for task $i \in V$, e.g., $E_i = \lceil \frac{t_i + \sum_{j \in P_i^*} t_j}{\bar{c}} \rceil$
L_i	Latest station for task $i \in V$, e.g., $L_i = \bar{m} + 1 - \lceil \frac{t_i + \sum_{j \in F_i^*} t_j}{\bar{c}} \rceil$
$KD(KP)$	Set of definite (probable) stations, i.e., $KD = \{1, \dots, \underline{m}\}$, and $KP = \{\underline{m} + 1, \dots, \bar{m}\}$
K	Set of all possible stations, i.e., $K = KD \cup KP$
FS_i	Set of stations to which task $i \in V$ is feasibly assignable, i.e., $FS_i = \{E_i, E_i + 1, \dots, L_i\}$
FT_k	Set of tasks which are feasibly assignable to station $k \in K$, i.e., $FT_k = \{i \in V \mid k \in FS_i\}$
A_i	Set of tasks that cannot be assigned to the station to which task i is assigned, e.g., $A_i = \{j \in V \mid FS_j \cap FS_i = \phi\}$
$F_i^F(P_i^F)$	Set of tasks which may directly follow (precede) task i in a station load in forward direction, i.e., $F_i^F = \{j \in V - (F_i^* - F_i) - P_i^* - A_i - \{i\}\}$ and $P_i^F = \{j \in V \mid i \in F_j^F\}$
$F_i^B(P_i^B)$	Set of tasks which may directly follow (precede) task i in a station load in backward direction, i.e., $F_i^B = \{j \in V - F_i^* - A_i\}$ and $P_i^B = \{j \in V \mid i \in F_j^B\}$
τ_{ij}	Forward setup time from task $i \in V$ to task $j \in F_i^F$
μ_{ij}	Backward setup time from task $i \in V$ to task $j \in F_i^B$

latter makes the problem (much) harder than the SALBP. The problem formulation is also more complicated than that of the SALBP.

The mathematical formulation proposed by Andrés et al. (2008) does not distinguish forward from backward setup times. Thus, the only available formulation for the SUALBSP is the mathematical formulation presented by Scholl et al. (2013). Before explaining this formulation, we direct your attention to Table 1 for the general parameters of the SUALBSP. The cycle time and the number of stations are denoted by c and m, respectively. c is a parameter for the SUALBSP-1, and m is a parameter for the SUALBSP-2. So, we have $c = \bar{c} = \underline{c}$ for the SUALBSP-1, and $m = \bar{m} = \underline{m}$ for the SUALBSP-2.

Scholl et al. (2013) formulate the SUALBSP-1 as a mixed integer linear programming formulation using the following sets of decision variables.

x_{ik} : binary variable with value 1, iff task $i \in V$ is assigned to station $k \in FS_i$
y_{ij} : binary variable with value 1, iff task $i \in V$ is direct predecessor of task $j \in F_i^F$ in the same station load
w_{ij} : binary variable with value 1, iff task $i \in V$ is last and $j \in F_i^B$ is first task in the same station load
z_i : continuous variable for encoding the number of the station task $i \in V$ is assigned to
f_i : continuous variable for start time of task $i \in V$ (relative to launch time at the first station)

It is noteworthy to mention that the model also includes a secondary objective to get station times with minimum setups. For the sake of simplicity, we just consider the original objective of the formulation as the basic model. See Appendix for different ways of formulating this secondary objective. The simplified formulation (excluding the secondary objective) can be expressed as follows. In this formulation, c is a parameter which denotes the cycle time, and M is a big-M parameter, i.e., $M = n \cdot c$.

Scholl et al. (2013) formulation (SF):

$$\text{SF: } \min z_n \tag{1}$$

$$\text{subject to: } \sum_{k \in FS_i} x_{ik} = 1 \qquad \forall i \in V \tag{2}$$

$$\sum_{k \in FS_i} k \cdot x_{ik} = z_i \qquad \forall i \in V \tag{3}$$

$$\sum_{j \in F_i^F} y_{ij} + \sum_{j \in F_i^B} w_{ij} = 1 \qquad \forall i \in V \tag{4}$$

$$\sum_{i \in P_j^F} y_{ij} + \sum_{i \in P_j^B} w_{ij} = 1 \qquad \forall j \in V \tag{5}$$

$$\sum_{i \in V} \sum_{j \in F_i^B} w_{ij} = z_n \tag{6}$$

$$z_j - z_i \le M \cdot (1 - y_{ij}) \text{ and } z_i - z_j \le M \cdot (1 - y_{ij}) \qquad \forall i \in V \text{ and } j \in F_i^F \tag{7}$$

$$z_j - z_i \le M \cdot (1 - w_{ij}) \text{ and } z_i - z_j \le M \cdot (1 - w_{ij}) \qquad \forall i \in V \text{ and } j \in F_i^B \tag{8}$$

$$f_j \ge f_i + t_i \qquad \forall (i, j) \in E \tag{9}$$

$$f_i \ge c \cdot (z_i - 1) \qquad \forall i \in V \tag{10}$$

$$f_i + (t_i + \tau_{ij}) + M \cdot (y_{ij} - 1) \le f_j \qquad \forall i \in V \text{ and } j \in F_i^F \tag{11}$$

$$f_i + (t_i + \mu_{ij}) + M \cdot (w_{ij} - 1) \le c \cdot z_i \qquad \forall i \in V \text{ and } j \in F_i^B \tag{12}$$

$$y_{ij} \in \{0, 1\} \qquad \forall i \in V \text{ and } j \in F_i^F \tag{13}$$

$$w_{ij} \in \{0, 1\} \qquad \forall i \in V \text{ and } j \in F_i^B \tag{14}$$

$$x_{ik} \in \{0, 1\} \qquad \forall i \in V \text{ and } k \in FS_i \qquad (15)$$

$$z_i \geq 0 \qquad \forall i \in V \qquad (16)$$

$$f_i \geq 0 \qquad \forall i \in V. \qquad (17)$$

In SF, the objective function (1) minimizes the index of the station to which the unique terminal node n is assigned. Since the terminal task is supposed to be the successor of all other tasks, the objective function (1) minimizes the required number of stations. As is described in Scholl et al. (2013), if no unique terminal node exists in the original precedence graph, then a dummy sink node is introduced and defined as the successor of the original terminal nodes. Moreover, the execution time for the dummy terminal node is set to c and all its setup times to zero. After solution, if a dummy node has been defined, the actual number of stations is calculated by $m = z_n - 1$.

Constraint set (2) ensures that each task is assigned to exactly one station. Constraint set (3) captures the station to which task i is assigned. The constraint sets (4) and (5) ensure that each task is followed and preceded by exactly one other task. Constraint set (6) guarantees that the number of positive w_{ij} variables is equal to the number of stations. According to the authors, constraint sets (4)–(6) assure that in each cycle exactly one of the relations is a backward setup. Constraint sets (7) and (8) are added so as to make sure that each relation is established between tasks of the same station. In other words, variables w_{ij} and y_{ij} are forced to be zero when $z_i \neq z_j$. Constraint set (9) ensures that precedence relations among the tasks are fulfilled. Constraint sets (10)–(12) set the start time of each task in the time interval $[c \cdot (z_i - 1), c \cdot z_i - t_i]$. In other words, constraint set (10) resets the start time of the first task in terms of the station index, and constraint set (11) sets the start time of its successors in the same station. The impact of backward setup times on the station time is considered in constraint set (12), where the station time is forced not to exceed the cycle time. Constraint sets (13)–(17) define the decision variables of the problem.

As is seen, Scholl et al. (2013) formulated the SUALBSP-1 in an intelligent manner that the number of binary variables and constraints is bounded by $O(n^2)$. Nevertheless, the formulation is incomplete. We explain the issue by the following counterexample.

Counterexample Figure 1 shows a precedence graph representing a simple instance of the problem with $n = 6$ tasks (nodes) and task times as node weights ($t \geq 1$). Suppose that $\mu_{23} = \mu_{32} = \mu_{42} = \varepsilon > 0$, and all other setup times are zero. If the cycle time is considered as $c = 2t$, then SF results in the solution shown in Table 2.

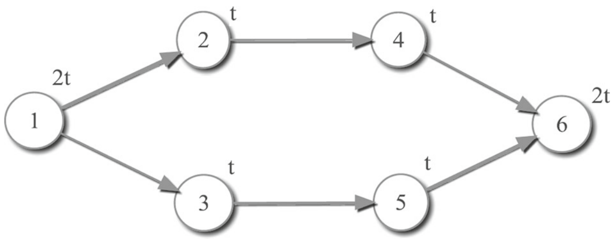


Fig. 1 The precedence graph of the counterexample

Table 2 Solution to the counterexample provided by SF

i	1	2	3	4	5	6
z_i	1	2	2	2	2	4
f_i	0	$2t$	$2t$	$3t$	$3t$	$6t$
$\{j \mid y_{ij} = 1\}$	ϕ	{4}	{5}	ϕ	ϕ	ϕ
$\{j \mid w_{ij} = 1\}$	{1}	ϕ	ϕ	{3}	{2}	{6}

According to this solution, the station time for station 2 is more than two times bigger than the specified cycle time. Moreover, the formulation offers 4 stations as the optimal solution. Nonetheless, it is not hard to see that the optimal number of stations is $m = 5$. Thus, SF does not ensure feasibility or optimality of its solutions. This is due to the fact that each station must include exactly one combination (i, j) of tasks with $w_{ij} = 1$, but this is not guaranteed by SF. To resolve such deficiency, we propose to add the following constraints into the formulation.

$$o_{ik} \leq x_{ik} \qquad \forall i \in V \text{ and } k \in FS_i \tag{18}$$

$$\sum_{j \in F_i^B} w_{ij} \leq \sum_{k \in FS_i} o_{ik} \qquad \forall i \in V \tag{19}$$

$$\sum_{i \in FT_k} o_{ik} \leq 1 \qquad \forall k \in K \tag{20}$$

$$o_{ik} \geq 0 \qquad \forall i \in V \text{ and } k \in FS_i \tag{21}$$

According to these constraint sets, continuous variable o_{ik} gets value 1 if i is the last task of station k and 0 otherwise. Constraint set (20) in conjunction with constraint sets (4)–(6) ensures that each station includes at most one combination (i, j) of tasks with $w_{ij} = 1$.

3 New formulations

In this section, three new formulations for the SUALBSP are developed by introducing new decision variables. Two of these formulations are station-based models while the other is a schedule-based one. Unlike the station-based models, in the schedule-based model there is no index for denoting the stations. Indeed, in the schedule-based formulation the problem is completely formulated as a scheduling problem. We first consider the previous formulation (SF) so as to provide a more effective mathematical formulation.

3.1 The first station-based formulation (FSBF)

A cursory look reveals that SF requires a large number of disjunctive constraints, i.e., constraints with big-M parameters. In the following, a new formulation (FSBF) is developed in which the disjunctive constraints of SF are modified.

Scholl et al. (2013) employ a continuous decision variable f_i to represent the start time of task $i \in V$ relative to launch time at the first station. We introduce a new continuous decision variable s_i to denote the start time of task $i \in V$ in the station to which this task is assigned. Also, binary variable u_k is used, which gets value 1 if and only if any task is assigned to station $k \in KP$. Using the new decision variables, FSBF for the SUALBSP-1 (FSBF-1) is expressed as follows.

$$\text{FSBF-1: } \min \sum_{k \in KP} u_k + \underline{m} \tag{22}$$

subject to: (2), (3), (4), (5), (13), (14), (15), (16), (18), (19), (20), (21), and:

$$z_j - z_i \leq (L_j - E_i) \cdot (1 - y_{ij}) \text{ and } z_i - z_j \leq (L_i - E_j) \cdot (1 - y_{ij}) \quad \forall i \in V \text{ and } j \in F_i^F \tag{23}$$

$$z_j - z_i \leq (L_j - E_i) \cdot (1 - w_{ij}) \text{ and } z_i - z_j \leq (L_i - E_j) \cdot (1 - w_{ij}) \quad \forall i \in V \text{ and } j \in F_i^B \tag{24}$$

$$\sum_{i \in V} \sum_{j \in F_i^B} w_{ij} - \sum_{k \in KP} u_k \geq \underline{m} \tag{25}$$

$$\sum_{i \in FT_k} t_i \cdot x_{ik} \leq c \quad \forall k \in KD \tag{26}$$

$$\sum_{i \in FT_k} t_i \cdot x_{ik} \leq c \cdot u_k \quad \forall k \in KP \tag{27}$$

$$s_i + c \cdot (z_i - z_j) + t_i + \tau_{ij} \cdot y_{ij} \leq s_j \quad \forall (i, j) \in E \tag{28}$$

$$s_i + (t_i + \tau_{ij}) + (c + \tau_{ij}) \cdot (y_{ij} - 1) \leq s_j \quad \forall i \in V \text{ and } j \in F_i^F \backslash F_i \tag{29}$$

$$s_i + t_i + \sum_{j \in F_i^B} \mu_{ij} \cdot w_{ij} \leq c \quad \forall i \in V \tag{30}$$

$$u_{k+1} \leq u_k \quad \forall k \in KP \backslash \{\bar{m}\} \tag{31}$$

$$s_i \geq 0 \quad \forall i \in V \tag{32}$$

In FSBF-1, the objective function (22) minimizes the number of stations. Constraint sets (23) and (24) are equivalent to constraint sets (7) and (8) in which the big-M parameter is modified. Constraint (25) in conjunction with constraint sets (4), (5), and (20) ensures that each station includes at most one combination (i, j) of tasks with $w_{ij} = 1$. Note that constraint set (6) has been replaced with constraint (25). Knapsack constraint sets (26) and (27) are added so as to strengthen the formulation. In addition, constraint set (27) imposes a relationship between the binary variables x_{ik} and u_k. Precedence relations are fulfilled by constraint set (28). This will be demonstrated by Proposition 1. In fact, constraint sets (28)–(30) are used instead of constraint sets (9), (11), and (12), respectively. These substitutions both reduce the number of constraints and alleviate the impact of disjunctive constraints. Constraint set (31) defeats the symmetry in the formulation by removing any empty station between two consecutive stations. These symmetry breaking constraints reduce the solution time by lessening the number of searched nodes in the Branch and Bound (B&B) tree. Finally, constraint set (32) makes sure that the continuous variables s_i are non-negative.

Proposition 1 *Constraint set* (28) *guarantees that precedence relations are established between the tasks.*

Proof Suppose that tasks $j \in V$ and $i \in P_j$ have been assigned to the same station ($z_i = z_j$). According to constraint set (28), when $z_i = z_j$ the execution of task $j \in V$ will begin at some time after task $i \in P_j$. Now, suppose that $z_i \neq z_j$ which in turn conveys that $y_{ij} = 0$. If $z_i > z_j$, then constraint set (28) implies that $s_j \geq s_i + t_i + c$, which gives a contradiction [see constraint set (30)]. Thus, for each $i \in P_j$ we always have $z_i \leq z_j$. □

Note that FSBF-1 minimizes the number of stations without finding/creating a unique sink node in the precedence graph. Moreover, the introduction of variables s_i enabled us to use a more suitable value as the big-M parameter in constraint set (29). The value of $\underline{m}$ and $\bar{m}$ in the formulation can be easily set to $\underline{m} = \lceil t_{\text{sum}}/c \rceil$ and $\bar{m} = n$, respectively.

FSBF-1 can be simply applied to the SUALBSP-2 by a few modifications. If the number of stations is given by m, then FSBF for the SUALBSP-2 (FSBF-2) is stated as follows. In this formulation, c is a decision variable.

$$\text{FSBF-2: } \min c \tag{33}$$

subject to: (2), (3), (4), (5), (13), (14), (15), (16), (18), (19), (20), (21), (23), (24), (26), (30), (32), and:

$$\sum_{i \in V} \sum_{j \in F_i^B} w_{ij} \geq m \tag{34}$$

$$s_i + \bar{c} \cdot (z_i - z_j) + t_i + \tau_{ij} \cdot y_{ij} \leq s_j \qquad \forall (i, j) \in E \tag{35}$$

$$s_i + (t_i + \tau_{ij}) + (\bar{c} + \tau_{ij}) \cdot (y_{ij} - 1) \leq s_j \qquad \forall i \in V \text{ and } j \in F_i^F \setminus F_i \tag{36}$$

$$\underline{c} \leq c \leq \bar{c} \tag{37}$$

In FSBF-2, the objective function (33) minimizes the value of the cycle time. Constraint sets (25), (28), and (29) in FSBF-1 have been replaced with constraint sets (34)–(36), respectively. Constraint set (37) determines the domain of the decision variable c.

In FSBF-2, $\underline{c}$ and $\bar{c}$ are the lower bound and upper bound on the cycle time, respectively. The lower bound on the cycle time can be easily calculated as:

$$\underline{c} = \max\left\{t'_{\max}, \left\lceil \frac{t_{\text{sum}}}{m} \right\rceil\right\} \tag{38}$$

where $t'_{\max} = \max_{i\in V}(t_i + \mu_{ii})$. The upper bound on the cycle time can be calculated both theoretically and heuristically. Since the theoretical upper bounds are usually weak, it seems better to calculate the upper bounds via heuristic methods. One may employ a simple priority rule-based procedure for calculating $\bar{c}$ in the SUALBSP-2 formulations.

3.2 The second station-based formulation (SSBF)

We now present another station-based formulation for the SUALBSP. Unlike the FSBF, the number of variables in the SSBF is bounded by $O(n^3)$. This formulation is a modified version of the binary program which is developed by Andrés et al. (2008). In their formulation, the forward and backward setup times are not distinguished. Besides, the number of variables and constraints in that formulation are bounded by $O(n^3)$ and $O(n^4)$, respectively. SSBF is a better formulation by only $O(n^2)$ constraints. In this formulation, the following new decision variables are used.

g_{ijk} : binary variable with value 1, iff task i is performed immediately before task j in station k

h_{ijk} : binary variable with value 1, iff task i is the last and task j is the first task in station k

r_i : ordering variable representing the priority of task i in a sequence (a continuous variable)

The SSBF for the SUALBSP-1 (SSBF-1) is as follows.

$$\text{SSBF-1:} \ \min \sum_{k\in KP} u_k + \underline{m} \tag{22}$$

subject to:

(2), (3), (15), (16), (31), and:

$$\sum_{j\in(FT_k\cap F_i^F)} g_{ijk} + \sum_{j\in(FT_k\cap F_i^B)} h_{ijk} = x_{ik} \qquad \forall i \in V \text{ and } k \in FS_i \tag{39}$$

$$\sum_{i\in(FT_k\cap P_j^F)} g_{ijk} + \sum_{i\in(FT_k\cap P_j^B)} h_{ijk} = x_{jk} \qquad \forall j \in V \text{ and } k \in FS_j \tag{40}$$

$$\sum_{i\in FT_k} \sum_{j\in(FT_k\cap F_i^B)} h_{ijk} = 1 \qquad \forall k \in KD \tag{41}$$

$$\sum_{i\in FT_k} \sum_{j\in(FT_k\cap F_i^B)} h_{ijk} = u_k \qquad \forall k \in KP \tag{42}$$

$$r_i + 1 + (n - |F_i^*| - |P_j^*|) \cdot \left(\sum_{k \in (FS_i \cap FS_j)} g_{ijk} - 1 \right) \leq r_j \quad \forall i \in V \text{ and } j \in F_i^F \tag{43}$$

$$r_i + 1 \leq r_j \qquad \forall (i, j) \in E \tag{44}$$

$$z_i \leq z_j \qquad \forall (i, j) \in E \tag{45}$$

$$\sum_{i \in FT_k} t_i \cdot x_{ik} + \sum_{i \in FT_k} \sum_{j \in (FT_k \cap F_i^F)} \tau_{ij} \cdot g_{ijk} + \sum_{i \in FT_k} \sum_{j \in (FT_k \cap F_i^B)} \mu_{ij} \cdot h_{ijk} \leq c \qquad \forall k \in KD \tag{46}$$

$$\sum_{i \in FT_k} t_i \cdot x_{ik} + \sum_{i \in FT_k} \sum_{j \in (FT_k \cap F_i^F)} \tau_{ij} \cdot g_{ijk} + \sum_{i \in FT_k} \sum_{j \in (FT_k \cap F_i^B)} \mu_{ij} \cdot h_{ijk} \leq c \cdot u_k \qquad \forall k \in KP \tag{47}$$

$$\sum_{i \in FT_k \setminus \{j\}} x_{ik} \leq (n - \underline{m} + 1) \cdot (1 - h_{jjk}) \qquad \forall k \in K \text{ and } j \in FT_k \tag{48}$$

$$g_{ijk} \in \{0, 1\} \qquad \forall k \in K,\ i \in FT_k,\ j \in (FT_k \cap F_i^F) \tag{49}$$

$$h_{ijk} \in \{0, 1\} \qquad \forall k \in K,\ i \in FT_k,\ j \in (FT_k \cap F_i^B) \tag{50}$$

$$|P_i^*| + 1 \leq r_i \leq n - |F_i^*| \qquad \forall i \in V \tag{51}$$

In SSBF-1, the objective function (22) minimizes the number of stations. Constraint sets (39) and (40) ensure that a task in station k is followed and preceded by exactly one other task in the cyclic sequence of this station. According to constraint sets (41) and (42), in each cycle exactly one of the relations is a backward setup. Constraint sets (43) and (44) establish the precedence relations among the tasks within each station. Note that the constraint set (43) is inactive if tasks i and j are assigned to different stations [$g_{ijk} = 0$ for each $k \in (FS_i \cap FS_j)$]. So, we add the constraint set (45) to make sure that the precedence relations among the tasks of different stations are held. Knapsack constraints (46) and (47) guarantee that no station time exceeds the cycle time. Constraints (48) ensure that only task j is allocated to station k when $h_{jjk} = 1$. Constraint sets (49)–(51) specify domain of the variables.

The SSBF for the SUALBSP-2 (SSBF-2) can be simply expressed as follows.

$$\text{SSBF-2: } \min c \tag{33}$$

$$\begin{aligned} \text{subject to: } & (2),\ (3),\ (15),\ (16),\ (37), \\ & (39),\ (40),\ (41),\ (43),\ (44),\ (45), \\ & (46),\ (48),\ (49),\ (50),\ (51). \end{aligned}$$

As is seen, SSBF-2 and SSBF-1 differ only in a few constraints. In comparison to the binary program proposed by Andrés et al. (2008), the size of SSBF is drastically smaller. Unfortunately, the number of binary variables in this formulation is still high. In Sect. 4, we seek to reduce the number of binary variables of the SSBF by preprocessing approaches. However, the size complexity of this formulation makes it difficult to be solved even if we solve just the LP-relaxations of it.

3.3 The schedule-based formulation (SCBF)

There are different ways to formulate the SUALBSP as a scheduling problem. Due to the existence of the precedence relations, it seems better to use a binary decision variable q_{ij} which gets value 1 if and only if task i is executed before task j. In this case, $i \in P_j^*$ conveys that $q_{ij} = 1$ and $q_{ji} = 0$. Therefore, it is enough to define the binary variable q_{ij} only for each $i \in V$ and $j \in V \backslash (P_i^* \cup F_i^* \cup \{i\})$. Using the new decision variable q_{ij}, the schedule-based formulation for the SUALBSP-1 (SCBF-1) is expressed as follows. The number of variables and constraints in the SCBF-1 is bounded by $O(n^2)$.

$$\text{SCBF-1:} \ \min \sum_{i \in V} \sum_{j \in F_i^B} w_{ij} \tag{52}$$

$$\text{subject to: } (4),\ (5),\ (13),\ (14),\ (30),\ (32),\ (44),\ (51),\ \text{and:}$$

$$s_i + (t_i + \tau_{ij}) + (c + \tau_{ij}) \cdot (y_{ij} - 1) \le s_j \qquad \forall i \in V \text{ and } j \in F_i^F \tag{53}$$

$$q_{ij} + q_{ji} = 1 \qquad \forall i \in V \text{ and } j \in V \backslash (P_i^* \cup F_i^*) \text{ with } i < j \tag{54}$$

$$r_i + 1 + (n - |F_i^*| - |P_j^*|) \times (q_{ij} - 1) \le r_j \qquad \forall i \in V \text{ and } j \in V \backslash (P_i^* \cup F_i^*) \text{ with } i \ne j \tag{55}$$

$$r_j - 1 + (n - |F_j^*| - |P_i^*| - 2) \times (y_{ij} - 1) \le r_i \qquad \forall i \in V \text{ and } j \in F_i^F \tag{56}$$

$$y_{ij} \le q_{ij} \qquad \forall i \in V \text{ and } j \in F_i^F \backslash F_i^* \tag{57}$$

$$w_{ji} \le q_{ij} \qquad \forall i \in V \text{ and } j \in P_i^B \backslash F_i^* \text{ with } i \ne j \tag{58}$$

$$q_{ij} \in \{0, 1\} \qquad \forall i \in V \text{ and } j \in V \backslash (P_i^* \cup F_i^*) \text{ with } i \ne j \tag{59}$$

First note that the output of SCBF-1 is a schedule of all the tasks, and the variable r_i represents the position of task i in the schedule. In this formulation, the objective function (52) minimizes the number of stations. Constraint set (53) is equivalent to the constraint set (29), which is written for each $i \in V$ and $j \in F_i^F$. Logical constraint set (54) establishes a relationship between two tasks i and j which are not related by precedence. Constraint set (55) together with constraint sets (44) and (54) enforces a *transitive relation* between the decision variables q_{ij}. In other words, for three tasks i, j, and v, these constraints guarantee that $q_{iv} = 1$ when $q_{ij} = q_{jv} = 1$ (see Esmaeilbeigi et al. 2015a). Constraint set (56) in conjunction with constraint sets (55) and (57) implies that $r_j = r_i + 1$ when $y_{ij} = 1$. In fact, these constraints make sure that a forward setup can only occur between two consecutive tasks in the schedule. According to constraint sets (57) and (58), both forward and backward setups occur along the schedule. Constraint set (59) defines domain of the variables q_{ij}.

In Sect. 2, we realize that without some additional constraints, SF may result in impractical solutions. For this reason, in SSBF-1, we included constraints (41) and (42). Proposition 2 states that the schedule-based formulation does not require such constraints.

Proposition 2 *In the schedule-based formulation, each station load is explicitly indicated by a unique pair* (i, j) *of tasks with* $w_{ij} = 1$.

Proof The assignment constraints (4) and (5) ensure that $H = \{(i, j) : y_{ij} = 1 \text{ or } w_{ij} = 1\}$ defines either one or a collection of Hamiltonian cycles (tours). Each tour represents a station load in the SUALBSP. We show that every tour includes exactly one arc (i, j) with $w_{ij} = 1$. Consider any feasible solution of the SCBF where all the tasks are sorted into one schedule. By constraint sets (57) and (58) we know that forward setups only occur in a forward direction and backward setups in a backward direction of this schedule. Clearly, a tour cannot be created without at least one arc (i, j) with $w_{ij} = 1$. On the other hand, forward setups only occur between consecutive tasks of the schedule which implies that a tour has just one arc (i, j) with $w_{ij} = 1$. □

The SCBF for the SUALBSP-2 (SCBF-2) is as follows.

$$\text{SCBF-2:}\ \min c \qquad (33)$$

$$\text{subject to: } (4),\ (5),\ (13),\ (14),\ (30),\ (32),\ (37),\ (44),\ (51),\ (54),\ (55),\ (56),\ (57),\ (58),\ (59),\ \text{and}:$$

$$s_i + (t_i + \tau_{ij}) + (\bar{c} + \tau_{ij}) \cdot (y_{ij} - 1) \le s_j \quad \forall i \in V \text{ and } j \in F_i^F \tag{60}$$

$$\sum_{i \in V} \sum_{j \in F_i^B} w_{ij} = m \tag{61}$$

In SCBF-2, the objective function (33) minimizes the value of the cycle time. Constraint set (60) is equivalent to the constraint set (36), which is written for each $i \in V$ and $j \in F_i^F$. Constraint (61) assures that exactly m stations are established.

The main disadvantage of the SCBF is that it usually consumes large amounts of memory when a standard linear solver is used. The reason is that the size of the active node list in the B&B tree grows rapidly. However, this formulation is capable of generating good solutions quickly.

4 Improvements

The SUALBSP-2 can be considered as m TSPs in which the set of cities for each TSP is unknown (in the SUALBSP-1 even the number of TSPs is a decision variable). Because each TSP (with a given set of cities) is NP-hard by itself, the NP-hard nature of the SUALBSP is comprehensible. Therefore, it is not unexpected that exact solution approaches to the SUALBSP like mathematical formulations may only solve small up to median instances of the problem. In this Section, we try to alleviate the computational complexity of the mathematical formulations by a couple of preprocessing and cutting techniques.

4.1 Incompatible tasks detection

The concept of earliest and latest station [introduced by Patterson and Albracht (1975)] is a typical preprocessing approach for the assembly line balancing formulations with assignment variables like x_{ik}. The basis of this preprocessing is the use of the precedence relations to determine lower bounds and upper bounds on the station index to which a task can be assigned. The definition of sets FS_i and FT_k is based on this preprocessing. We now present another preprocessing based on the precedence relations.

If we know, in advance, that two specific tasks cannot be assigned to the same station, then we do not need to generate the corresponding variables in the FSBF, SSBF, and SCBF. Proposition 3 allows us to recognize such incompatible tasks.

Proposition 3 *Consider three tasks $i, j, v \in V$ such that $i \in P_j^*$, and $v \in (F_i^* \cap P_j^*)$. In this case, tasks i, j, and v are assigned to the same station if and only if tasks i and j belong to the same station.*

Proof The proof is trivial and is omitted. □

Based on Proposition 3, if we define parameter d_{ij} as

$$d_{ij} = d_{ji} = \begin{cases} t_i + t_j + \sum_{v \in (F_i^* \cap P_j^*)} t_v & \forall i \in V, j \in F_i^* \\ 0 & \text{Otherwise.} \end{cases} \tag{62}$$

Then, tasks $i, j \in V$ cannot be assigned to the same station if $d_{ij} > c$. Employing this idea, a preprocessing can be performed which reduces the number of variables y_{ij}, w_{ij}, h_{ijk}, and g_{ijk}. To implement this preprocessing, one can easily set

$$A_i = \{j \in V \mid d_{ij} > c \text{ or } (FS_j \cap FS_i) = \phi\}$$

A similar approach is also applied by Scholl et al. (2010) in a so-called *extended task time incrementing rule*. Note that Scholl et al. (2013) just consider $(FS_j \cap FS_i) = \phi$ as a condition for restricting elements of sets P_i^F, P_i^B, F_i^F, and F_i^B. The efficacy of this approach is highly dependent on the upper bound on the number of stations.

Parameter d_{ij} defined by (62) can be further improved (increased) by taking setup times into account. To this purpose, we first define sets B_i and R_{ij} as follows.

$$\begin{cases} B_i = \{j \in V \mid (FS_j \cap FS_i) = \phi\} & \forall i \in V \\ R_{ij} = \{i, j\} \cup \{v \in V \mid v \in (F_i^* \cap P_j^*)\} & \forall i \in V \text{ and } j \in F_i^* \backslash B_i \end{cases}$$

Observe that by Proposition 3, if tasks i and j are assigned to the same station, then R_{ij} is a subset of that station load. As already mentioned, in the SUALBSP, a (modified) triangle inequality is fulfilled (see Scholl et al. (2013) for further discussions). These triangular inequalities can be stated as follows.

$$\tau_{iv} + t_v + \tau_{vj} \geq \tau_{ij} \quad \text{and} \quad \tau_{iv} + t_v + \mu_{vj} \geq \mu_{ij} \quad \forall i, j, v \in V \tag{63}$$

These triangle inequalities imply that the station time is not decreased if an additional task is added to the station load. This means that we can provide a lower bound on the time of a station to which tasks $i \in V$ and $j \in F_i^* \backslash B_i$ are assigned by just considering the setup times for the tasks $v \in R_{ij}$. To develop such a lower bound, we define some additional parameters as follows.

$$\begin{cases} \delta_{ivj} = \min_{v' \in (R_{ij} \cap F_v^F)}(\tau_{vv'}) & \forall i \in V,\ j \in F_i^* \backslash B_i \text{ and } v \in R_{ij} \backslash \{j\} \\ \delta'_{ivj} = \min_{v' \in (R_{ij} \cap P_v^F)}(\tau_{v'v}) & \forall i \in V,\ j \in F_i^* \backslash B_i \text{ and } v \in R_{ij} \backslash \{i\} \end{cases}$$

According to these definitions, $\max(\sum_{v \in R_{ij} \backslash \{j\}} \delta_{ivj}, \sum_{v \in R_{ij} \backslash \{i\}} \delta'_{ivj})$ is a lower bound on the forward setup times of the station. Thus, by employing the triangular inequalities, we can improve the parameter d_{ij} as follows.

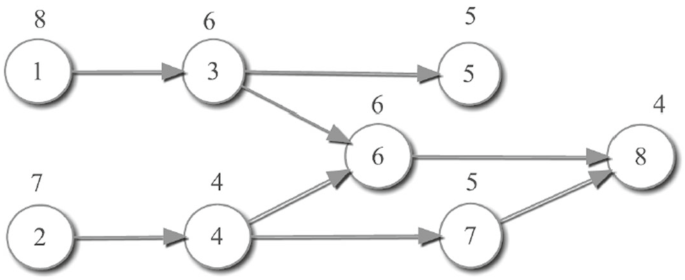
$$d_{ij} = d_{ji} = \begin{cases} \mu_{ji} + \sum_{v \in R_{ij}} t_v + \max(\sum_{v \in R_{ij} \backslash \{j\}} \delta_{ivj}, \sum_{v \in R_{ij} \backslash \{i\}} \delta'_{ivj}) & \forall i \in V,\ j \in F_i^* \backslash B_i \\ 0 & \text{Otherwise.} \end{cases} \tag{64}$$

After calculating the value of this parameter, set A_i is determined as $A_i = B_i \cup \{j \in V \mid d_{ij} > c\}$. We now give an example to illustrate the above preprocessing.

Example 1 The precedence graph and the forward and backward setup times for the example presented by Scholl et al. (2013) are depicted in Fig. 2. The cycle time in this example is $c = 20$. Note that we manipulated the original figure by putting $\mu_{1,7}$, $\tau_{1,7}$, $\tau_{7,1}$, and $\tau_{7,3}$ to zero (these setups are also possible).

The values of the parameters R_{ij}, δ_{ivj}, δ'_{ivj}, and d_{ij} for three pairs (i, j) of tasks are given in Table 3.

From Table 3, we see that two tasks $i = 1$ and $j = 3$ cannot be assigned to the same station because $22 = d_{1,3} > 20$. Note that we cannot set $(i, j) = (1, 3)$ as incompatible tasks if we calculate the parameter d_{ij} through (62). This is also true for the tasks $(i, j) = (3, 8)$ and $(i, j) = (4, 8)$.



τ	1	2	3	4	5	6	7	8
1	-	1	2	1	-	-	0	-
2	1	-	3	2	1	-	-	-
3	-	3	-	1	2	3	1	-
4	1	-	1	-	1	2	0	-
5	-	1	-	1	-	1	1	2
6	-	-	-	-	1	-	2	3
7	0	-	0	-	1	2	-	1
8	-	-	-	-	2	-	-	-

μ	1	2	3	4	5	6	7	8
1	4	5	-	3	-	-	0	-
2	3	4	1	-	3	-	-	-
3	6	7	4	5	-	-	5	-
4	5	6	3	4	5	-	-	-
5	4	5	2	3	4	5	3	2
6	3	4	1	2	3	4	2	-
7	5	6	3	4	5	6	4	-
8	6	7	4	5	6	7	5	4

Fig. 2 Data of the example presented by Scholl et al. (2013)

Table 3 The value of the parameters in the preprocessing for the presented example

(i, j)	(1,3)		(3,8)			(4,8)			
$v \in R_{ij}$	$v = 1$	$v = 3$	$v = 3$	$v = 6$	$v = 8$	$v = 4$	$v = 6$	$v = 7$	$v = 8$
δ_{ivj}	$\tau_{1,3}$	–	$\tau_{3,6}$	$\tau_{6,8}$	–	$\tau_{4,7}$	$\tau_{6,7}$	$\tau_{7,8}$	–
δ'_{ivj}	–	$\tau_{1,3}$	–	$\tau_{3,6}$	$\tau_{6,8}$	–	$\tau_{4,6}$ or $\tau_{7,6}$	$\tau_{4,7}$	$\tau_{7,8}$
d_{ij}	22		26			27			

For the SUALBSP-2, where the cycle time is a decision variable, A_i is achieved by replacing c with $\bar{c}$. Unfortunately, the resulted preprocessing is not very effective when $\bar{c}$ is large. Therefore, we propose another scheme which presents the above preprocessing in the form of valid inequalities. These valid inequalities can be stated as inequalities (65).

$$c + \bar{c} \cdot (z_j - z_i) \geq d_{ij} \qquad \forall i \in V \quad \text{and} \quad j \in F_i^* \backslash B_i \tag{65}$$

According to these inequalities, when tasks $i \in V$ and $j \in F_i^* \backslash B_i$ are assigned to the same station, d_{ij} gives a lower bound on the cycle time. Note that valid inequalities (65) cannot be used in the SCBF.

4.2 Extension of earliest/latest stations

Another preprocessing approach can be implemented for the SSBF. In this preprocessing, we generalize the concept of earliest and latest stations when two specific tasks are assigned to the same station. Let a_{ij}^{F} and b_{ij}^{F} be the earliest and latest station for tasks $i \in V$ and $j \in F_i^F$ when they are assigned to the same station by an occurring forward setup. Also, suppose that a_{ij}^{B} and b_{ij}^{B} be the earliest and latest station for tasks $i \in V$ and $j \in F_i^B$ when they are assigned to the same station by an occurring backward setup. We calculate these parameters as follows.

$$\begin{cases} a_{ij}^{F} = \left\lceil \dfrac{t_i + \tau_{ij} + t_j + \sum_{v \in (P_i^* \cup P_j^*) \setminus \{i,j\}} t_v}{c} \right\rceil & \forall i \in V \text{ and } j \in F_i^F \\ b_{ij}^{F} = \bar{m} + 1 - \left\lceil \dfrac{t_i + \tau_{ij} + t_j + \sum_{v \in (F_i^* \cup F_j^*) \setminus \{i,j\}} t_v}{c} \right\rceil & \forall i \in V \text{ and } j \in F_i^F \end{cases}$$

$$\begin{cases} a_{ij}^{B} = \left\lceil \dfrac{t_i + \mu_{ij} + t_j + \sum_{v \in (P_i^* \cup P_j^*) \setminus \{i,j\}} t_v}{c} \right\rceil & \forall i \in V \text{ and } j \in F_i^B \\ b_{ij}^{B} = \bar{m} + 1 - \left\lceil \dfrac{t_i + \mu_{ij} + t_j + \sum_{v \in (F_i^* \cup F_j^*) \setminus \{i,j\}} t_v}{c} \right\rceil & \forall i \in V \text{ and } j \in F_i^B \end{cases}$$

Now, using these parameters, variables g_{ijk} and h_{ijk} should be defined and generated as follows.

$$\begin{aligned} g_{ijk} &\in \{0,1\} \quad \forall i \in V, j \in F_i^F \text{ and } k \in \left[a_{ij}^{F}, b_{ij}^{F}\right] \\ h_{ijk} &\in \{0,1\} \quad \forall i \in V, j \in F_i^B \text{ and } k \in \left[a_{ij}^{B}, b_{ij}^{B}\right] \end{aligned}$$

In FSBF, this idea can be imposed on the formulation by including the following valid inequalities.

$$\sum_{k=E_i}^{a_{ij}^{F}-1} x_{ik} + \sum_{k=b_{ij}^{F}+1}^{L_i} x_{ik} + y_{ij} \leq 1 \qquad \forall (i,j) \in N_1^F \tag{66}$$

$$\sum_{k=a_{ij}^{F}}^{b_{ij}^{F}} x_{ik} \geq y_{ij} \qquad \forall (i,j) \in N_2^F \tag{67}$$

$$\sum_{k=E_i}^{a_{ij}^{B}-1} x_{ik} + \sum_{k=b_{ij}^{B}+1}^{L_i} x_{ik} + w_{ij} \leq 1 \qquad \forall (i,j) \in N_1^B \tag{68}$$

$$\sum_{k=a_{ij}^B}^{b_{ij}^B} x_{ik} \geq w_{ij} \qquad \forall (i,j) \in N_2^B \tag{69}$$

In which sets N_1^F, N_2^F, N_1^B, and N_2^B are defined as follows:

$$\begin{cases} N_1^F = \{(i,j) \mid i \in V, j \in F_i^F, (L_i - E_i) \leq 2 \cdot (b_{ij}^F - a_{ij}^F)\} \\ N_2^F = \{(i,j) \mid i \in V, j \in F_i^F, (L_i - E_i) > 2 \cdot (b_{ij}^F - a_{ij}^F)\} \\ N_1^B = \{(i,j) \mid i \in V, j \in F_i^B, (L_i - E_i) \leq 2 \cdot (b_{ij}^B - a_{ij}^B)\} \\ N_2^B = \{(i,j) \mid i \in V, j \in F_i^B, (L_i - E_i) > 2 \cdot (b_{ij}^B - a_{ij}^B)\} \end{cases}$$

These valid inequalities restrict the possible stations to which task i (and therefore task j) can be assigned. Note that considering $N_1^F = N_2^F = \{(i,j) \mid i \in V, j \in F_i^F\}$, valid inequalities (66) and (67) are identical. We distinguish the sets in such a form so as to reduce the number of non-zero elements (NNZ) of the coefficient matrix as much as possible. For example, suppose that $a_{ij}^F = b_{ij}^F = L_i = 4$ and $E_i = 2$. In this case, valid inequalities (66) and (67) are written as follows.

$$(66): \; x_{i,2} + x_{i,3} + y_{ij} \leq 1 \rightarrow NNZ = 3$$
$$(67): \; x_{i,4} \geq y_{ij} \rightarrow NNZ = 2$$

Since $x_{i,2} + x_{i,3} + x_{i,4} = 1$, the above inequalities are equivalent. However, constraint set (66) adds more non-zero elements to the formulation than the constraint set (67).

4.3 Valid inequalities

The following sets of logical constraints are valid inequalities for the relevant formulations.

$$y_{ij} + y_{ji} \leq 1 \qquad \forall i \in V \text{ and } j \in (F_i^F \cap P_i^F) \tag{70}$$

$$\sum_{j \in F_i^B} w_{ij} + \sum_{j \in P_i^B} w_{ji} + w_{ii} \leq 1 \qquad \forall i \in V \tag{71}$$

Valid inequalities (70) say that between variables y_{ij} and y_{ji} only one of them can get value 1. Although similar inequalities can be considered for w_{ij} variables, valid inequalities (71) dominate those inequalities.

We now present a new set of valid inequalities which significantly improves the LP-relaxation of the FSBF and SCBF. As earlier mentioned, station time in the SUALBSP is the total execution time of tasks assigned to a station plus their sequence-dependent setup times. The idle time is then defined as the difference between the cycle time and the station time. Let I be the total idle time, and S be the total setup time. It is not difficult to see that in the SUALBSP the relation $S + I + t_{\text{sum}} = m \cdot c$ holds. In other words, we have:

$$\sum_{i\in V}\sum_{j\in F_i^F}\tau_{ij}\cdot y_{ij}+\sum_{i\in V}\sum_{j\in F_i^B}\mu_{ij}\cdot w_{ij}+t_{\text{sum}}+I=m\cdot c \tag{72}$$

Therefore, the following inequalities are valid for any type of the SUALBSP.

$$\sum_{i\in V}\sum_{j\in F_i^F}\tau_{ij}\cdot y_{ij}+\sum_{i\in V}\sum_{j\in F_i^B}\mu_{ij}\cdot w_{ij}+t_{\text{sum}}\le m\cdot c \tag{73}$$

The left-hand side of these inequalities gives a lower bound on the line capacity. These inequalities can be directly used in the FSBF-2 and SCBF-2. To incorporate these inequalities into the FSBF-1, one needs to substitute m with $\underline{m}+\sum_{k\in KP}u_k$ as follows:

$$\sum_{i\in V}\sum_{j\in F_i^F}\tau_{ij}\cdot y_{ij}+\sum_{i\in V}\sum_{j\in F_i^B}\mu_{ij}\cdot w_{ij}-c\cdot\sum_{k\in KP}u_k\le c\cdot\underline{m}-t_{\text{sum}} \tag{74}$$

Also, for the SCBF-1, these inequalities can be used in the following form:

$$\sum_{i\in V}\sum_{j\in F_i^F}\tau_{ij}\cdot y_{ij}+t_{\text{sum}}\le\sum_{i\in V}\sum_{j\in F_i^B}(c-\mu_{ij})\cdot w_{ij} \tag{75}$$

It is noteworthy to say that without these valid inequalities the SCBF cannot achieve satisfactory results because its lower bounds would be extremely unfavorable.

5 Computational results

To evaluate performance of the SUALBSP-1 formulations, we conducted an extensive computational study on the SBF data sets. According to Scholl et al. (2013) some drawbacks of the other data sets have been removed from the SBF data sets. There are two versions of the SBF data sets, SBF1 and SBF2. In this paper, we use SBF2 data sets in which the optimal solutions are available (see Scholl et al. 2013 for further details). These data sets, which are given for the SUALBSP-1, can be found in the website http://alb.mansci.de. The size of the setup times in the SBF data sets is specified with a factor α. This parameter influences the average value of setup times in terms of the average task times. Simply put, a high value of α represents higher setup time values. Four different data sets are obtained by setting α to the values 0.25, 0.50, 0.75, and 1.00.

We used C++ programming language to code the formulations and preprocessing techniques and used CPLEX 12.4 to solve the formulations. All the experiments were run on a computer with a single-core 2.5 GHz Intel processor and 4.0 GB RAM. The time limit was set to 1800 seconds. To make a better comparison, three classes of instances denoted by $C1$, $C2$, and $C3$ are considered. $C1$ includes very small instances up to 21 tasks, $C2$ includes small instances with 25–30 tasks, and $C3$ includes medium instances with 32–58 tasks. In total, $C1$, $C2$, and $C3$ have 108, 112, and 176 instances, respectively.

To quantify the performance of each formulation, the value of CPU time (CPU), relative optimality gap (GAP), and the number of search nodes (NODE), all reported by CPLEX, are presented. We sometimes compare the lower bounds achieved by the formulations with the optimal solutions using $\text{GAP}_{\text{LB}} = \frac{\text{optimal-LB}_{\text{formulation}}}{\text{optimal}} \times 100$. We show the average value (Avg), the median (Mdn), and the maximum (Max) of these quantities. Where suitable, the percentages of instances in which at least one feasible solution was found (Feas) and the percentages of the instances that are solved to optimality (Opt) are also reported.

5.1 Performance of the new formulations in comparison to SF

To show that the new formulations perform better than the previous one (SF), we compare them on $C1$. The results of solving all four formulations are depicted in Table 4. In the FSBF-1, SSBF-1, and SCBF-1, all the preprocessing approaches have been performed so as to reduce the number of binary variables of them. We also added the capacity constraints (74) and (75), respectively, to FSBF-1 and SCBF-1 to achieve their best performance. Note that the SSBF-1 does not require the constraint set (74) as constraint sets (46) and (47) dominate it.

According to our experiments on C1, C2, and C3, the preprocessing introduced in Sect. 4.1 can eliminate 26.53 % of variables w_{ij} on average. Note that this preprocessing rarely eliminates forward related variables, i.e., y_{ij} (in FSBF-1 and SCBF-1) and g_{ijk} (in SSBF-1). The reason is that in this case, the parameter d_{ij} in the preprocessing approach gets a small value which hardly exceeds the cycle time.

Table 4 The comparison of all the formulations on $C1$

α	Formulation	CPU			GAP			NODE		
		Avg	Mdn	Max	Avg	Mdn	Max	Avg	Mdn	Max
0.25	SF	86.44	0.69	1800.00	0.41	0.00	11.11	13,951	460	157,214
	FSBF-1	0.52	0.30	1.64	0.00	0.00	0.00	3	0	50
	SSBF-1	1.27	0.64	4.66	0.00	0.00	0.00	0	0	0
	SCBF-1	87.43	0.30	1800.09	0.46	0.00	12.50	19,685	0	332,720
0.50	SF	96.03	1.15	1800.00	0.41	0.00	11.11	15,849	1270	245,217
	FSBF-1	0.85	0.47	8.45	0.00	0.00	0.00	44	0	878
	SSBF-1	1.31	0.52	5.31	0.00	0.00	0.00	0	0	0
	SCBF-1	22.63	0.33	308.00	0.00	0.00	0.00	8211	0	125,614
0.75	SF	208.49	2.02	1800.03	1.64	0.00	20.00	33,420	2075	441,364
	FSBF-1	1.22	0.45	17.50	0.00	0.00	0.00	126	0	3372
	SSBF-1	1.40	0.64	5.72	0.00	0.00	0.00	0	0	0
	SCBF-1	8.64	0.30	193.24	0.00	0.00	0.00	2846	0	56107
1.00	SF	217.22	1.22	1799.99	0.53	0.00	14.24	39,397	1193	389,363
	FSBF-1	0.96	0.33	9.10	0.00	0.00	0.00	104	0	2053
	SSBF-1	1.40	0.47	5.41	0.00	0.00	0.00	0	0	0
	SCBF-1	2.11	0.26	24.43	0.00	0.00	0.00	1027	0	18,164

Table 5 Comparison of the LP-relaxation bounds using $C1$, $C2$, and $C3$

α	FSBF-1			SSBF-1			SCBF-1		
	Avg	Mdn	Max	Avg	Mdn	Max	Avg	Mdn	Max
0.25									
GAP_{LB}	3.01	0.00	20.00	2.72	0.00	20.00	4.00	0.00	20.00
CPU	0.80	0.30	3.78	25.37	4.04	140.62	0.06	0.05	0.16
0.50									
GAP_{LB}	3.01	0.00	20.00	2.72	0.00	20.00	4.00	0.00	20.00
CPU	0.75	0.28	3.59	24.55	4.67	144.90	0.06	0.05	0.24
0.75									
GAP_{LB}	3.01	0.00	20.00	2.55	0.00	20.00	4.00	0.00	20.00
CPU	0.69	0.27	3.31	23.30	4.76	122.96	0.06	0.05	0.18
1.00									
GAP_{LB}	3.01	0.00	20.00	2.55	0.00	20.00	4.00	0.00	20.00
CPU	0.73	0.28	3.74	21.83	3.70	125.34	0.06	0.05	0.16

From Table 4, we can see that FSBF-1 and SSBF-1 impressively outperform SF [completed by constraint sets (18)–(21)] in terms of the computational time, optimality gap, and the number of searched nodes. Usually, by an increase in the value of α, the solution times of the formulations increase, but SCBF-1 proves better performance. The SSBF-1 surprisingly solves all the problem instances at the root node which demonstrates its tightness.

5.2 Performance of the LP-relaxations

To obtain lower bounds on the SUALBSP-1, we can solve LP-relaxations (LP) of the proposed formulations. In Table 5 we compare the LP bounds (LB) of the new formulations with the optimal solutions. Since the number of stations is an integer, each fractional solution is rounded up to achieve a better LB. The proposed preprocessing techniques, introduced in Sects. 4.1 and 4.2, are also implemented to improve the LBs. According to the table, $\text{LB}_{\text{SCBF-1}} \leq \text{LB}_{\text{FSBF-1}} \leq \text{LB}_{\text{SSBF-1}}$, which means, on average, SSBF-1 provides better LBs on the chosen data.

It is also important to compare the LP solution times of the formulations. From the table, the LPs of SCBF-1 are solved very fast in comparison to the other two formulations. Because of its high size complexity, the LP-relaxations of the SSBF-1 require a lot of time to be solve.

5.3 The comparison between FSBF-1 and SSBF-1

In Tables 6 and 7, we, respectively, report results of solving FSBF-1 and SSBF-1 on $C2$. From these tables, we can interpret that the main weakness of the FSBF-1 is that it cannot find feasible solutions quickly. However, SSBF-1 finds at least one feasible

Table 6 The results of FSBF-1 on $C2$

α	CPU			GAP			Feas (%)	Opt (%)
	Avg	Mdn	Max	Avg	Mdn	Max		
0.25	372.11	80.74	1800.04	1.52	0.00	11.11	100.00	85.71
0.50	706.87	427.62	1800.04	2.47	0.00	16.67	89.29	71.43
0.75	970.56	1008.13	1800.05	4.84	0.00	25.00	92.86	62.96
1.00	1173.04	1800.00	1800.05	6.76	0.00	54.55	82.14	46.43
Avg	805.65	829.12	1800.05	3.78	0.00	25.81	91.07	66.63

Table 7 The results of SSBF-1 on $C2$

α	CPU			GAP			Feas (%)	Opt (%)
	Avg	Mdn	Max	Avg	Mdn	Max		
0.25	219.29	108.02	1800.04	0.40	0.00	11.11	100.00	96.43
0.50	332.33	151.19	1800.03	0.58	0.00	9.09	100.00	92.86
0.75	380.97	159.29	1800.04	0.77	0.00	12.50	100.00	92.86
1.00	396.80	178.00	1800.10	1.59	0.00	44.44	100.00	96.43
Avg	332.35	149.13	1800.05	0.84	0.00	19.29	100.00	94.65

Table 8 The comparison of FSBF-1 and SSBF-1 on $C3$

α	Formulation	CPU			GAP			GAP$_{\text{LB}}$			Feas (%)	Opt (%)
		Avg	Mdn	Max	Avg	Mdn	Max	Avg	Mdn	Max		
0.25	FSBF-1	1269.20	1800.03	1800.51	3.90	0.00	20.00	1.46	0.00	9.09	47.73	34.09
	SSBF-1	1302.47	1800.08	1815.84	6.04	0.00	50.00	1.66	0.00	7.41	52.27	36.36
0.50	FSBF-1	1442.75	1800.03	1800.08	6.04	0.00	25.00	1.50	0.00	7.41	36.36	25.00
	SSBF-1	1357.44	1800.08	1800.16	8.47	0.00	65.38	1.70	0.00	9.09	47.73	34.09
0.75	FSBF-1	1545.34	1800.02	1800.61	3.46	0.00	16.67	1.63	0.00	9.09	29.55	20.45
	SSBF-1	1338.84	1800.08	1817.81	15.03	0.00	82.76	1.50	0.00	7.41	50.00	34.09
1.00	FSBF-1	1532.62	1800.02	1800.07	8.12	10.56	25.00	1.71	0.00	9.09	36.36	15.91
	SSBF-1	1260.11	1800.07	1800.17	9.76	0.00	86.36	1.53	0.00	7.41	60.47	44.19

solution for every instance. The SSBF-1 also provides optimal solutions for almost 95 % of the instances. These formulations can be better compared via Table 8 where their results on $C3$ are reported. This table also indicates that the SSBF-1 solves more problem instances to optimality than the FSBF-1. In the column GAP$_{\text{LB}}$, the relative gaps between the optimal solutions and the lower bounds (achieved by the formulations after 1800 s) are reported. According to this column, both formulations produce almost equal lower bounds. It also shows that the FSBF-1's difficulty is to detect feasible solutions not to find good lower bounds. Although the SSBF-1 performs better than the FSBF-1 in small/medium problem instances, the size complexity of

Table 9 The impact of valid inequalities (66)–(71) on FSBF-1 using $C2$

α	GAP			CPU			GAP_{LB}			Feas (%)		
	$L1$	$L2$	$L3$	$L1$	$L2$	$L3$	$L1$	$L2$	$L3$	$L1$	$L2$	$L3$
0.25	1.79	1.50	1.96	372.11	468.02	575.76	0.32	0.00	0.32	100.00	100.00	100.00
0.50	2.86	4.11	4.75	706.87	880.60	811.40	0.83	0.32	0.34	89.29	100.00	96.43
0.75	4.87	8.94	5.22	970.56	1025.56	1147.52	0.71	0.83	1.13	96.43	92.86	85.71
1.00	6.97	11.15	8.38	1173.04	1096.73	1172.88	0.00	0.35	0.77	82.14	92.86	78.57
Avg	4.01	6.29	4.81	805.65	867.73	926.89	0.47	0.38	0.64	91.96	96.43	90.18

this formulation could be a severe shortcoming in solving larger instances. Even the LP-relaxations of this formulation cannot be solved in a reasonable time if the number of tasks exceeds 100.

5.4 Performance of the valid inequalities

To evaluate constraint sets (66)–(71), we added them in the FSBF-1 and tested them on $C2$. The data (average) are reported in Table 9. $L1$ in the table denotes for the original formulation with the valid inequalities (74). Note that results of $L1$ in solving instances of $C1$ are presented in Table 4. $L2$ represents $L1$ with additional logical inequalities (70) and (71). Also, $L3$ denotes for $L1$ with additional valid inequalities (66)–(69).

From the table, it is inferred that these valid inequalities are not quite effective in solving the FSBF-1. In other words, applying them resulted in increasing the optimality gap and the computational time in our test instances. This is mainly because the computational time highly depends on finding both good (global) lower bound and upper bound early on the search in the B&B tree. Our computational experiments show that using these valid inequalities can improve the value of the objective function at root node in the B&B tree. This implies that we can obtain a better lower bound early on the search. However, CPLEX struggles in finding good feasible solutions (upper bounds) after adding these valid inequalities. As a consequence, increasing the optimality gap and the computational time in our test instances is not surprising.

5.5 Performance of the SCBF-1

From Table 5, we understand that LP-relaxations of the SCBF-1 are solved very fast. Therefore, the solver can explore a large number of nodes in the B&B tree quickly. On the other hand, the formulation has problems with finding good lower bounds and pruning this huge amount of active nodes. Consequently, the size of the active node list grows rapidly which frequently results in an Out of Memory error. To be able to examine the formulation, we set a time limit of 5 min and a node limit of 100,000 nodes for this formulation. Using these settings, we solved the SCBF-1 and FSBF-1 on $C2$ and $C3$. The results are displayed in Table 10 where we compared

Table 10 Results achieved by FSBF-1 and SCBF-1 after 5 min

α	FSBF-1				SCBF-1			
	GAP_{UB}			Feas (%)	GAP_{UB}			Feas (%)
	Avg	Mdn	Max		Avg	Mdn	Max	
0.25	3.84	0.00	25.12	58.33	11.32	0.00	37.50	97.26
0.50	5.45	0.00	21.43	41.67	7.71	0.00	33.33	97.22
0.75	10.19	0.00	83.24	33.33	6.49	0.00	30.00	100.00
1.00	4.04	0.00	44.44	27.78	7.20	0.00	36.36	98.61
Avg	5.61	0.00	39.52	40.28	8.16	0.00	34.27	98.27

the feasible solutions provided by these formulations. We calculated the relative gap between the upper bounds achieve by the formulations with the optimal solutions by $GAP_{UB} = \frac{UB_{formulation} - optimal}{UB_{formulation}} \times 100$. As the table shows, the quality of the solutions to the SCBF-1 after 5 min is reasonably good. Note that this formulation can provide feasible solutions for more than 98 % of the instances, but FSBF-1 can detect feasible solutions for only 40 %.

6 Conclusion

In this paper, three different mathematical formulations are presented for the SUALBSP. To strengthen these formulations, several possible improvements in the form of valid inequalities and preprocessing approaches are proposed. One of these formulations is able to generate good feasible solutions within a few minutes. This formulation is a schedule-based model, where the SUALBSP is formulated as a scheduling problem. Since this formulation requires a huge amount of memory during the solution procedure, usually a time/node limit should be imposed on the formulation. The other two formulations presented in this paper for the SUALBSP are station-based models. They are modifications of the previous formulations presented in the literature which can be used for providing lower bounds on the problem. The number of binary variables and constraints in the first station-based formulation is bounded by $O(n^2)$. The second station-based formulation provides good lower bounds, but it has $O(n^3)$ variables which renders it useless when the size of the problem is large. The first station-based formulation, on the other hand, can find good lower bounds for solving larger instances of the problem.

In this research, we focused on the SUALBSP-1 in our computational experiments. One promising direction for future research is to evaluate formulations of the SUALBSP-2. The impact of minimizing the total setup time (as a secondary objective) can also be considered in computational experiments. In addition, the performance of the formulations could be examined on other data sets to provide a better understanding of the formulations. Finally, the experimental results in the paper indicate the high combinatorial nature of the problem. Although mathematical formulations are the first step in solving the SUALBSP to optimality, still there is a need for more efficient exact methods.

Appendix

The secondary objective

Scholl et al. (2013) consider the objective function (76) and constraint sets (77) and (78) to embed a secondary objective into their formulation.

$$\min \; z_n + \varepsilon \cdot \sum_{k \in K} T_k \tag{76}$$

$$(f_i - c \cdot (z_i - 1)) + t_i + \sum_{j \in F_i^B} \mu_{ij} \cdot w_{ij} \le T_k + M \cdot (1 - x_{ik}) \quad \forall i \in V \text{ and } k \in FS_i \tag{77}$$

$$T_k \ge 0 \qquad \forall k \in K \tag{78}$$

In this model, T_k is a continuous variable representing the station time for station k. In addition, ε is a sufficiently small number, i.e., $\varepsilon = 1/(c \cdot (n+1))$. The secondary objective minimizes the total setup time by minimizing the sum of the station times. Constraint set (77) in conjunction with the objective function (76) captures the station time of station k. Constraint set (78) states non-negativity of the variables T_k. As is seen, including the secondary objective (76) requires addition of $O(n^2)$ disjunctive constraints.

One other way to incorporate the secondary objective into the formulation is to use the following objective function and constraints:

$$\min \; z_n + \varepsilon \cdot \sum_{i \in V} T_i' \tag{79}$$

$$(f_i - c \cdot (z_i - 1)) + t_i + \sum_{j \in F_i^B} (c + \mu_{ij}) \cdot w_{ij} \le T_i' + c \qquad \forall i \in V \tag{80}$$

$$T_i' \ge 0 \qquad \forall i \in V \tag{81}$$

According to this model, if task i is the last task in its station, then variable T_i' gets the value of the station time of that station to which task i is assigned, and zero otherwise. This method requires only n additional constraints.

Still, the secondary objective can be included into the formulation more effectively. Let idle time be defined as the difference between the cycle time and the station time. If I be the total idle time, and S be the total setup time, the relation $S + I + t_{\text{sum}} = m \cdot c$ holds. So, for a given value of the line capacity, maximization of the total idle time is equivalent to minimization of the total setup time. Thus, the following scheme is proposed.

$$\min \; (\bar{m} \cdot c - t_{\text{sum}} + 1) \cdot z_n - I \tag{82}$$

$$\sum_{i \in V} \sum_{j \in F_i^F} \tau_{ij} \cdot y_{ij} + \sum_{i \in V} \sum_{j \in F_i^B} \mu_{ij} \cdot w_{ij} + t_{\text{sum}} + I \le c \cdot z_n \tag{83}$$

$$I \geq 0 \tag{84}$$

The objective function (82) minimizes the number of stations as the primary objective and the total setup time as the secondary objective. Because z_n takes an integer value and $I \in [0, \bar{m} \cdot c - t_{\text{sum}}]$, maximization of the total idle time (minimization of the total setup time) has no effect on optimization of the primary objective. This approach just needs one additional constraint.

References

Andrés C, Miralles C, Pastor R (2008) Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. Eur J Oper Res 187(3):1212–1223

Battaïa O, Dolgui A (2013) A taxonomy of line balancing problems and their solutionapproaches. Int J Prod Econ 142(2):259–277 (anticipation of risks impacts and industrial performance evaluation in distributed organizations life cycles)

Baybars İ (1986) A survey of exact algorithms for the simple assembly line balancing problem. Manag Sci 32(8):909–932

Becker C, Scholl A (2009) Balancing assembly lines with variable parallel workplaces: problem definition and effective solution procedure. Eur J Oper Res 199(2):359–374

Bowman EH (1960) Assembly-line balancing by linear programming. Oper Res 8(3):385–389

Dolgui A, Kovalev S, Kovalyov MY, Nossack J, Pesch E (2014) Minimizing setup costs in a transfer line design problem with sequential operation processing. Int J Prod Econ 151:186–194

Erel E, Sarin SC (1998) A survey of the assembly line balancing procedures. Prod Plan Control 9(5):414–434

Esmaeilbeigi R, Charkhgard P, Charkhgard H (2015a) Order acceptance and scheduling problems in two-machine flow shops: new mixed integer programming formulations. Eur J Oper Res 000:1–13. doi:10.1016/j.ejor.2015.11.036

Esmaeilbeigi R, Naderi B, Charkhgard P (2015b) The type e simple assembly line balancing problem: a mixed integer linear programming formulation. Comput Oper Res 64:168–177

Kucukkoc I, Zhang DZ (2015) Balancing of parallel u-shaped assembly lines. Comput Oper Res 64:233–244

Martino L, Pastor R (2010) Heuristic procedures for solving the general assembly line balancing problem with setups. Int J Prod Res 48(6):1787–1804

Pape T (2015) Heuristics and lower bounds for the simple assembly line balancing problem type 1: overview, computational tests and improvements. Eur J Oper Res 240(1):32–42

Pastor R, Ferrer L (2009) An improved mathematical program to solve the simple assembly line balancing problem. Int J Prod Res 47(11):2943–2959

Patterson JH, Albracht JJ (1975) Technical noteassembly-line balancing: Zero-one programming with fibonacci search. Oper Res 23(1):166–172

Scholl A, Boysen N, Fliedner M (2008) The sequence-dependent assembly line balancing problem. OR Spectr 30(3):579–609

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

Scholl A, Fliedner M, Boysen N (2010) Absalom: Balancing assembly lines with assignment restrictions. Eur J Oper Res 200(3):688–701

Sewell EC, Jacobson SH (2012) A branch, bound, and remember algorithm for the simple assembly line balancing problem. INFORMS J Comput 24(3):433–442

White WW (1961) Letter to the editorcomments on a paper by bowman. Oper Res 9(2):274–276

Yolmeh A, Kianfar F (2012) An efficient hybrid genetic algorithm to solve assembly line balancing problem with sequence-dependent setup times. Comput Ind Eng 62(4):936–945