Accepted Manuscript : https://doi.org/10.1016/j.ejor.2021.12.032

New Integer Optimization Models and an Approximate Dynamic Programming Algorithm for the Lot-sizing and Scheduling Problem with Sequence-dependent Setups

Younsoo Lee[a], Kyungsik Lee[a,*]

[a] *Department of Industrial Engineering, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul 08826, Republic of Korea*

Abstract

In this paper, we propose new integer optimization models for the lot-sizing and scheduling problem with sequence-dependent setups, based on the general lot-sizing and scheduling problem. To incorporate setup crossover and carryover, we first propose a standard model that straightforwardly adapts a formulation technique from the literature. Then, as the main contribution, we propose a novel optimization model that incorporates the notion of time flow. We derive a family of valid inequalities with which to compare the tightness of the models' linear programming relaxations. In addition, we provide an approximate dynamic programming algorithm that estimates the value of a state using its lower and upper bounds. Then, we conduct computational experiments to demonstrate the competitiveness of the proposed models and the solution algorithm. The test results show that the newly proposed time-flow model has considerable advantages compared with the standard model in terms of tightness and solvability. The proposed algorithm also shows computational benefits over the standard mixed integer programming solver.

Keywords: Production, Lot-sizing and scheduling problem, Integer optimization model, Sequence-dependent setup, Approximate dynamic programming algorithm

1. Introduction

The lot-sizing and scheduling problem (LSP) simultaneously determines the sizes of production lots and the production sequence within a given planning horizon. The objective is to minimize the total costs, including the production cost, inventory holding cost, backlog penalty cost, and setup cost. There have been many studies on this problem over recent decades, resulting in substantial advancements. However, as real-world manufacturing processes have a variety of unique characteristics and the features to be considered have been evolving, there remain outstanding issues not yet

*Corresponding author.
Email addresses: fifays@snu.ac.kr (Younsoo Lee), optima@snu.ac.kr (Kyungsik Lee)

completely resolved, and indeed, active research on optimization models and solution approaches tailored to solve real problems continues.

One of the important factors to consider in the LSP is setup activity. In certain manufacturing environments where it can be performed immediately at a small cost, the setup often is ignored when establishing the production plan. However, in cases where the setup takes a considerable amount of time or incurs substantial costs, it should be considered carefully in the planning stage. Moreover, in many real manufacturing processes, the setup is sequence-dependent; that is, its cost and time depend on both the item that was produced before and that which will be produced subsequently. It is known that sequence dependency makes the LSP much more difficult to solve (see Guimarães et al., 2014). In this study, we focused on solving the LSP with long and sequence-dependent setups.

LSPs with long and sequence-dependent setups are common in many manufacturing processes. In the fine-chemical process industry considered by Sung & Maravelias (2008), for instance, the setup takes a significant amount of time, as it includes several activities such as cleansing and testing. Similarly, in the flat-panel display manufacturing process studied by Lee & Lee (2020), the setup for highly automated equipment can take even longer than a day and occurs in a sequence-dependent manner. Certainly, in these manufacturing environments, the setup must be considered as an important factor in the planning stage.

There are several LSP models, which are broadly classified into big bucket models and small bucket models (see Copil et al., 2017, for a detailed review of these models). The bucket length of typical big bucket models is relatively long; therefore, several items can be produced within a single bucket. In small bucket models, contrastingly, the length of each bucket is relatively short, such that only one or at most two items can be produced within a bucket. One alternative model is the general lot-sizing and scheduling problem (GLSP, Fleischmann & Meyr, 1997), which is often called a hybrid of the big and small bucket models because it uses a two-level time structure. In the GLSP, the planning horizon is divided into multiple *macroperiods* the lengths of which are fixed. Each macroperiod is then divided into several *microperiods* the lengths of which are variables to be determined. External dynamics such as the arrival of the demand or inspection of the inventory levels are modeled at the end of the macroperiods, while internal dynamics such as the production amounts and the start/end of the setups are modeled within the microperiods.

When the setup takes a relatively long time, it can occur across consecutive time buckets, which situation is called *setup crossover*. In addition, *setup carryover*, which indicates the preservation of the setup state across time buckets, should be allowed in order to avoid unnecessary setups. As shown by Fiorotto et al. (2020), if setup crossover and carryover are not considered, the quality of the established production schedule can deteriorate significantly, or a feasible schedule may not be found even if it exists. Previous studies have discussed setup crossover and carryover and their consideration (Suerie, 2006; Belo-Filho et al., 2014; Fiorotto et al., 2017). Particularly, there has been much effort devoted to development of big bucket models that can consider various

characteristics such as nontriangular sequence-dependent setup as well as setup crossover and carryover, which has resulted in more complex models with a larger number of variables and constraints (e.g. Menezes et al., 2011; Clark et al., 2014; Mahdieh et al., 2018).

As pointed out by Almeder & Almada-Lobo (2011), GLSP-based models enable accurate modeling, as they incorporate the two-level time structure. For these models, unlike big bucket models, consideration of sequence-dependent setup is straightforward. Because the sequence of the microperiods is fixed, the production sequence within each macroperiod is naturally obtained (Camargo et al., 2012). Moreover, setup carryover can be expressed in such a way that the setup states of the last microperiod in the previous macroperiod and the first microperiod in the subsequent macroperiod are identical.

The modeling framework of the GLSP also is beneficial when the setup times are relatively long, because the maximum number of items that can be produced within a macroperiod is small, which leads to a smaller number of microperiods. Therefore, to deal with the long and sequence-dependent setups in this study, we focused on GLSP-based models. However, to the best of our knowledge, there is a lack of studies addressing setup crossover for GLSP-based models. Moreover, as pointed out by Almada-Lobo et al. (2015), introducing setup crossover into GLSP-based models is not straightforward.

Our contributions are as follows. We first propose a standard GLSP-based `(ST)` model that straightforwardly adapts a formulation technique from the literature to incorporate setup crossover. Then, as the main contribution, we propose a novel time-flow `(TF)` model that uses a set of decision variables representing the time flow for consideration of setup crossover. The `(TF)` model has considerable advantages over `(ST)` model in terms of tightness of linear programming (LP) relaxation and solvability with the standard mixed integer programming (MIP) solver. These advantages are demonstrated by both theoretical analysis with a family of valid inequalities and the results of computational experiments. In addition, we propose an approximate dynamic programming (ADP) algorithm that estimates the value of a state using its lower and upper bounds. The ADP algorithm showed computational benefits over the standard MIP solver. Also, its performance was revealed to be competitive with a state-of-the-art big bucket model.

The remainder of this paper is organized as follows. In Section 2, we review the literature related to both the problem and the solution algorithm that we consider. In Section 3, we present the optimization models and describe the differences among them by deriving a family of valid inequalities. In Section 4, we propose an ADP algorithm. In Section 5, we present the results of computational experiments. In Section 6, we conclude and look ahead to possible future extensions of the present work.

2. Literature Review

Relatively recent and comprehensive reviews of the LSP with sequence-dependent setups can be found in Guimarães et al. (2014) and Copil et al. (2017). Carvalho & Nascimento (2021) addressed parallel machine LSP with nontriangular sequence-dependent setups and setup carryover. The authors devised matheuristic algorithms by hybridizing mathematical programming and local search heuristics. Melega et al. (2020) studied a two-stage lot-sizing, scheduling and cutting stock problem in which the cutting decision is made in the first stage, while the lot-sizing and scheduling decisions are made in the second stage. Specifically, the authors, considering the sequence-dependent setups in both stages, proposed a heuristic algorithm combining a column generation approach and a relax-and-fix heuristic to deal with their integrated problem. Mahdieh et al. (2018) considered the LSP with nontriangular sequence-dependent setups, setup crossover, and carryover, proposing a multi-commodity-based big bucket model as an extension of the model presented in Clark et al. (2014).

The GLSP was first proposed by Fleischmann & Meyr (1997). As its name indicates, the GLSP can be regarded as a generalization of various models of LSP, because it has a two-level time structure consisting of macroperiods of fixed length and microperiods of variable length. The authors formally formulated the model, clarified the relationship between the GLSP and the existing models, and devised a heuristic algorithm that employs a local-search algorithm. Later, Koçlar & Süral (2005) indicated that the original GLSP provided by Fleischmann & Meyr (1997) has a minor limitation and corrected it. As indicated by Camargo et al. (2012), sequence-dependent setups can be incorporated into GLSP naturally because the sequence of microperiods is fixed.

Meyr (2000) proposed an extension of the GLSP that considers a sequence-dependent setup, namely, GLSPST. In addition, the heuristic algorithm of Fleischmann & Meyr (1997) was improved using a dual reoptimization technique. Meyr (2002) and Meyr & Mann (2013) considered the GLSPST with parallel machines, and solved it with heuristic algorithms. Recently, several studies have shown that the GLSPST can be extended to more general settings. Alem et al. (2018) considered the LSP with demand uncertainty and proposed a robust optimization model based on the GLSPST. Alipour et al. (2020) addressed the LSP with perishable products in the context of the food industry. They modeled the problem based on the GLSPST and proposed MIP-based heuristic algorithms.

Guimarães et al. (2014) pointed out that the GLSPST can be tightened using the well-known network flow reformulation presented by Wolsey (1997). Starting from the tightened GLSPST presented in Guimarães et al. (2014), we herein propose new GLSP-based models that can consider sequence-dependent setup, setup crossover, and carryover.

In fact, for consideration of setup crossover and carryover, various models have been proposed (see Fiorotto et al., 2020, for an extensive review). They are based on either big or small bucket models. Suerie & Stadtler (2003) proposed a big bucket model that incorporates setup carryover.

The authors then derived an extended formulation and presented families of valid inequalities. Later, this model was extended by Mohan et al. (2012) to further consider setup crossover. Suerie (2006) proposed two small bucket models that consider setup crossover.

Almada-Lobo et al. (2007) presented big bucket models with both sequence-dependent setup and setup carryover. These models treat a production sequence in a macroperiod as a connected path or cycle without any subtours. Based on these results, Menezes et al. (2011) further developed models to incorporate nontriangular setups by identifying certain types of admissible subtours. Further, the authors devised a new big bucket model that consider setup carryover and crossover simultaneously. It was shown by Fiorotto et al. (2017) that this model performs better than that proposed by Mohan et al. (2012). Therefore, we applied the formulation technique of Menezes et al. (2011) to the model of Guimarães et al. (2014), resulting in our first, standard model `(ST)`.

Like most optimization problems, the LSP can be reformulated as a dynamic programming (DP) problem. In fact, DP has been an essential solution approach for many lot-sizing problems. Since first being addressed by Wagner & Whitin (1958), many DP algorithms for lot-sizing problems have been proposed (e.g., Federgruen & Tzur, 1991). For a well-organized summary of the various algorithms for the class of LSP, including DP, readers are referred to Pochet & Wolsey (2006).

Except for certain special cases such as the uncapacitated problem, however, most real-world LSPs are difficult to solve via DP, because they often have a prohibitively large number of states. Therefore, obtaining an optimal solution of practical, large instances seems unpromising, because the number of states increases exponentially with the instance size. In order to avoid this issue, known as the "curse of dimensionality (Powell, 2007)", ADP algorithms have been widely employed. One of the most popular strategies of ADP is the *value function approximation* (Powell, 2016), which estimates the value of each state without recursive evaluation of future states.

There are many possibilities for approximating value functions for various problems. For instance, to solve multi-dimensional knapsack problems, Bertsimas & Demir (2002) used both the primal and dual bounds of the true state value, which were obtained by simple heuristic algorithms and LP relaxation, respectively. In addition, the authors proposed ADP algorithms with parametric and nonparametric approximations. Büyüktahtakın & Liu (2016) proposed various ADP algorithms for a single-item capacitated lot-sizing problem. Using the characteristics of the inventory cost function, they devised a direct-connection algorithm and a slope-check algorithm based on the sampling of the states. See Powell (2007, 2016) for several other successful applications of ADP. In this paper, we present an ADP algorithm that is similar to the approach presented by Bertsimas & Demir (2002). To obtain primal bounds, we used an LP-based fixing heuristic also known as *integer rounding heuristic* or *LP-rounding heuristic*. This heuristic has been widely used to solve LSPs. Maes et al. (1991) proposed several variants of the heuristic and compared their performance for multi-level problems. Alfieri et al. (2002) applied the LP-rounding heuristic to several LSP models and analyzed the differences.

3. Integer Optimization Models

Let $\mathcal{I} = \{1, \ldots, I\}$, $\mathcal{T} = \{1, \ldots, T\}$, $\mathcal{S} = \{1, \ldots, S\}$ be the sets of items, macroperiods, and microperiods, respectively. Throughout the exposition, $i, j \in \mathcal{I}$, $t \in \mathcal{T}$, $s \in \mathcal{S}$ are used for indices. Let hc_{it}, bc_{it}, pc_{it}, and d_{it} denote the unit inventory holding, backlogging, production cost, and demand for each item i and macroperiod t, respectively. Also, let sc_{ijt} be the cost incurred when setup occurs from item i to item j in macroperiod t. The production capacity for macroperiod t, given in time units, is denoted by K_t. The unit production time of item i is a_i, while the time required for setup from item i to item j is st_{ij}. We also define st_{ii} for item i and let the values be zero to represent the situation wherein the setup state is carried over. We let $\mathcal{S}_t \subset \mathcal{S}$ be the set of microperiods that are contained in macroperiod t. The first and the last microperiods within the macroperiod t are denoted by f^t and l^t, respectively. In other words, $\mathcal{S}_t = \{f^t, f^t + 1, \ldots, l^t\}$. Moreover, for microperiod s, we define $T(s)$ as the macroperiod that contains s; that is, $T(s) = t$ if and only if $s \in \mathcal{S}_t$. Let $start_t$ and end_t be the start/end time of macroperiod t; that is, $start_t = \sum_{k=1}^{t-1} K_t$ and $end_t = \sum_{k=1}^{t} K_t$ for all $t \in \mathcal{T}$.

Next, we define the decision variables. Let I_{it} and B_{it} be the inventory level and backlog amount of item i at the end of macroperiod t. The initial inventory and backlog for item i are denoted by I_{i0} and B_{i0}, which are assumed to be zero. The variable x_{is} represents the production amount of item i in microperiod s. Binary variable y_{is} is equal to one if item i is produced in microperiod s. For $i, j \in \mathcal{I}$ and $s \in \mathcal{S} \setminus \{1\}$, binary variable z_{ijs} is equal to one if the setup from item i to j occurs from microperiod $s-1$ to s. Variable z_{iis} represents the setup carryover of item i from microperiod $s-1$ to s (see Meyr & Mann, 2013). For notational convenience, we additionally define z_{ij1} for all $i, j \in \mathcal{I}$ and let their values be zero. For $i, j \in \mathcal{I}$ and $t \in \mathcal{T} \setminus \{1\}$, the binary variable q_{ijt} is equal to one if the setup from item i to item j crosses over from macroperiod $t-1$ to t. In this case, the setup time is split into $t-1$ and t. The continuous variable v_{ijt} represents the amount of the corresponding setup time distributed to $t-1$. It can be interpreted as the extra time borrowed from macroperiod $t-1$ to t for the setup crossover. Although there are no possibilities of setup crossover at the beginning of the first macroperiod and at the end of the last macroperiod, we define q_{ij1}, q_{itT+1}, v_{ij1}, and v_{ijT+1} for notational convenience and let their values be zero. All of the notations are summarized in Table 1.

3.1. Standard Model (ST)

We first provide the standard model (ST). This model is a generalization of the model presented by Guimarães et al. (2014) that can further consider setup crossover. After providing the model, we illustrate how setup crossover and carryover can be represented with the GLSP-based model. Later, this model is used to demonstrate the strength of our novel time-flow model (TF). The (ST) is as follows:

Table 1: Nomenclature

Sets and Indices	
$\mathcal{I}$	Set of items which are indexed by i and j; $i, j \in \mathcal{I} = \{1, \ldots, I\}$
$\mathcal{T}$	Set of macroperiods which are indexed by t; $t \in \mathcal{T} = \{1, \ldots, T\}$
$\mathcal{S}$	Set of microperiods which are indexed by s; $s \in \mathcal{S} = \{1, \ldots, S\}$
$\mathcal{S}_t$	Set of microperiods contained in a macroperiod t
Parameters	
hc_{it}	Inventory holding cost of item i in macroperiod t
bc_{it}	Backlog penalty cost of item i in macroperiod t
pc_{it}	Production cost of item i in macroperiod t
sc_{ijt}	Setup cost incurred when setup occurs from item i to item j in macroperiod t
d_{it}	Demand of item i in macroperiod t
K_t	Production capacity of macroperiod t given in time unit
a_i	Production time per unit of item i
st_{ij}	Time required for setup from item i to j. $st_{ii} = 0$
$T(s)$	Macroperiod that contains microperiod s, i.e., $T(s) = t \Leftrightarrow s \in \mathcal{S}_t$
f^t/l^t	First/last microperiod of macroperiod t, i.e., $\mathcal{S}_t = \{f^t, f^t + 1, \ldots, l^t\}$
$start_t$	Start time of macroperiod t, i.e., $start_t = \sum_{k=1}^{t-1} K_k$
end_t	End time of macroperiod t, i.e., $end_t = \sum_{k=1}^{t} K_k$
Variables	
I_{it}	Inventory level of item i at the end of macroperiod t, $I_{i0} = 0$
B_{it}	Backlog amount of item i at the end of macroperiod t, $B_{i0} = 0$
x_{is}	Production amount of item i in microperiod s
y_{is}	$= 1$ if item i is produced in microperiod s; $y_{is} = 0$, otherwise
z_{ijs}	$= 1$ if setup from item i to j occurs from microperiod $s-1$ to s; $z_{ijs} = 0$, otherwise
q_{ijt}	$= 1$ if setup from item i to j crosses over from macroperiod $t-1$ to t; $q_{ijt} = 0$, otherwise $q_{ij1} = q_{ijT+1} = 0$
v_{ijt}	Setup time split into macroperiod $t-1$, if $q_{ijt} = 1$; $v_{ijt} = 0$, otherwise $v_{ij1} = v_{ijT+1} = 0$

$$\text{minimize} \quad \sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}}\left(hc_{it}I_{it} + bc_{it}B_{it} + \sum_{s\in\mathcal{S}_t}\left(pc_{it}x_{is} + \sum_{j\in\mathcal{I}} sc_{ijt}z_{ijs}\right)\right) \tag{1a}$$

$$\text{subject to} \quad I_{it} - B_{it} + d_{it} = I_{it-1} - B_{it-1} + \sum_{s\in\mathcal{S}_t} x_{is} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{1b}$$

$$\sum_{i\in\mathcal{I}}\sum_{s\in\mathcal{S}_t}\left(a_i x_{is} + \sum_{j\in\mathcal{I}} st_{ji}z_{jis}\right) \le K_t + \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{I}}\left(v_{ijt} - v_{ijt+1}\right) \qquad \forall t \in \mathcal{T} \tag{1c}$$

$$a_i x_{is} \le K_t y_{is} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T}, s \in \mathcal{S}_t \tag{1d}$$

$$\sum_{i\in\mathcal{I}} y_{i1} = 1 \tag{1e}$$

$$y_{is} = \sum_{j\in\mathcal{I}} z_{jis} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S} \setminus \{1\} \tag{1f}$$

$$y_{is} = \sum_{j \in \mathcal{I}} z_{ijs+1} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S} \setminus \{S\} \quad \text{(1g)}$$

$$v_{ijt} \leq st_{ij} q_{ijt} \qquad \forall i, j \in \mathcal{I}, t \in \mathcal{T} \quad \text{(1h)}$$

$$q_{ijt} \leq z_{ijf^t} \qquad \forall i, j \in \mathcal{I}, t \in \mathcal{T} \quad \text{(1i)}$$

$$I_{it},\ B_{it},\ x_{is},\ y_{is},\ v_{ijt} \geq 0 \qquad \forall i \in \mathcal{I}, t \in \mathcal{T}, s \in \mathcal{S} \quad \text{(1j)}$$

$$q_{ijt}, z_{ijs} \in \{0, 1\} \qquad \forall i, j \in \mathcal{I}, t \in \mathcal{T}, s \in \mathcal{S} \quad \text{(1k)}$$

The objective function (1a) is the sum of the inventory holding, backlog penalty, production, and setup costs, which total is to be minimized. Constraints (1b) are balance equations between the inventory, backlog, demand, and production amounts. Constraints (1c) ensure that the sum of the production and setup times is less than or equal to the available capacity. The available capacity in t is computed in consideration of the time required for setup crossover. Constraints (1d) indicate that an item can be produced only if the corresponding setup occurs. Constraint (1e) represents the start of production in the first microperiod. Constraints (1f) and (1g) logically link the binary variables. They ensure that a setup from item i to j in microperiod s occurs if and only if i is produced in $s-1$ and j is produced in s. In other words, $z_{ijs} = 1$ if and only if $y_{is-1} = y_{js} = 1$. Constraints (1h) ensure that the setup time can be split only if the corresponding setup crossover occurs. The amount of split time is limited by the setup time. Constraints (1i) indicate that if the setup crossover occurs from $t-1$ to t, the item setup for the first microperiod of t can be determined. Constraints (1j) and (1k) ensure the domains of the variables. Note that the binary restriction of variable y is not necessary because it is implied by the constraints (1e)–(1g) and (1k).

With this model, a production plan can be represented as a network flow as shown in Figure 1. The bar shown in the upper part of Figure 1 represents a production plan wherein the productions

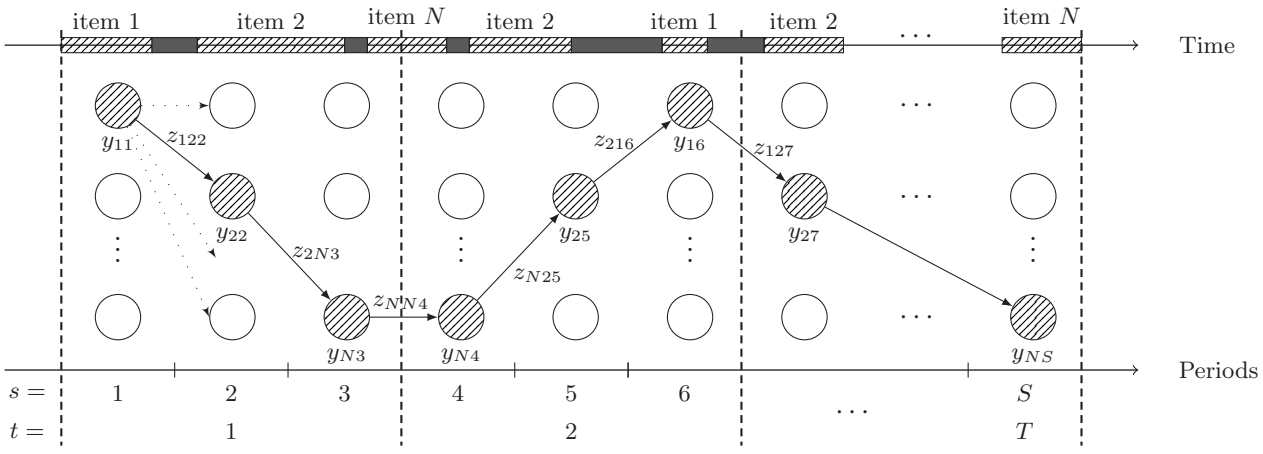

Figure 1: Illustration of the network flow corresponding to a production plan

and setups are indicated by the dashed and dark areas, respectively. This production plan is represented as a network flow in the lower part of Figure 1. The setup variables correspond to the arcs of the network, the flow balance equations of which correspond to the constraints (1e) – (1g). In this example, the setup state of item N is carried over from $s = 3$ to $s = 4$; that is, $z_{NN4} = 1$. Moreover, the setup between item 1 and item 2 is crossed over from $t = 2$ to $t = 3$; that is, $q_{123} = 1$.

3.2. Time-flow Model (TF)

We present the new time-flow model, which is denoted by (TF). For (TF), we introduce new variables and additional notations. Let us define a continuous variable w_{ijs} to represent the start time of the setup from item i to j in microperiod s, which can have a nonzero value only if $z_{ijs} = 1$. Note that, from the definition, $w_{ij1} = 0$ for all $i, j \in \mathcal{I}$. In addition, let r_{is} denote the idle time associated with item i in microperiod s. L_{ijs} and U_{ijs} denote the lower and upper bounds on the value of w_{ijs}, respectively. By adjusting these bounds, the model may or may not consider setup crossover, as shown later. The (TF) is as follows:

$$\text{minimize} \quad \sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}}\left(hc_{it}I_{it} + bc_{it}B_{it} + \sum_{s\in\mathcal{S}_t}\left(pc_{it}x_{is} + \sum_{j\in\mathcal{I}} sc_{ijt}z_{ijs}\right)\right) \tag{2a}$$

$$\text{subject to} \quad I_{it} - B_{it} + d_{it} = I_{it-1} - B_{it-1} + \sum_{s\in\mathcal{S}_t} x_{is} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \tag{2b}$$

$$\sum_{j\in\mathcal{I}}(w_{jis} + st_{ji}z_{jis}) + a_i x_{is} + r_{is} = \sum_{k\in\mathcal{I}} w_{iks+1} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S} \tag{2c}$$

$$a_i x_{is} + r_{is} \le K_t y_{is} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T}, s \in \mathcal{S}_t \tag{2d}$$

$$\sum_{i\in\mathcal{I}} y_{i1} = 1 \tag{2e}$$

$$y_{is} = \sum_{j\in\mathcal{I}} z_{jis} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S} \setminus \{1\} \tag{2f}$$

$$y_{is} = \sum_{j\in\mathcal{I}} z_{ijs+1} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S} \setminus \{S\} \tag{2g}$$

$$L_{ijs}z_{ijs} \le w_{ijs} \le U_{ijs}z_{ijs} \qquad \forall i, j \in \mathcal{I}, s \in \mathcal{S} \setminus \{1\} \tag{2h}$$

$$I_{it},\ B_{it},\ x_{is},\ r_{is},\ y_{is},\ w_{ijs} \ge 0 \qquad \forall i, j \in \mathcal{I}, t \in \mathcal{T}, s \in \mathcal{S} \tag{2i}$$

$$z_{ijs} \in \{0, 1\} \qquad \forall i, j \in \mathcal{I}, s \in \mathcal{S} \tag{2j}$$

The objective function (2a) and other constraints (2b) and (2e)—(2g) are defined as in (1a), (1b), and (1e)–(1g) of (ST), respectively. Constraints (2c) indicate that if an item i is set up in microperiod s from the previous item j ($z_{jis} = 1$), the sum of its start time (w_{jis}), setup time (st_{ji}), production time ($a_i x_{is}$), and idle time (r_{is}) is equal to the start time of the next setup to another item k (w_{iks+1}). Constraints (2c) can be regarded as the time-flow balance equations. Constraints (2d) indicate that an item can be produced only if the corresponding setup occurs. Constraints (2h) ensure that w_{ijs} cannot have a nonzero value unless the corresponding setup occurs. If the

corresponding setup occurs, its start time is bounded by L_{ijs} and U_{ijs}. Constraints (2i) and (2j) ensure the domains of the variables.

Now, we show how to set the values of L_{ijs} and U_{ijs}. If $s \in \mathcal{S} \setminus \{1\}$ and $s = f^{T(s)}$, the setup z_{ijs} can be crossed over. In this case, the setup start time w_{ijs} should be within the range $[start_{T(s-1)}, end_{T(s-1)}]$, while the setup end time $w_{ijs} + st_{ij}$ should be within the range $[start_{T(s)}, end_{T(s)}]$. Assuming that $st_{ij} \leq K_t$ for all $i, j \in \mathcal{I}$ and $t \in \mathcal{T}$, the following holds: $start_{T(s-1)} \leq start_{T(s)} - st_{ij}$ and $end_{T(s-1)} = start_{T(s)} \leq end_{T(s)} - st_{ij}$. Therefore, we can set $L_{ijs} = start_{T(s)} - st_{ij}$ and $U_{ijs} = start_{T(s)}$ for all $s \in \mathcal{S} \setminus \{1\}$ such that $s = f^{T(s)}$. On the contrary, when $s \neq f^{T(s)}$, there is no chance of setup crossover as both $s - 1$ and s belong to the same macroperiod. In this case, we can set $L_{ijs} = start_{T(s)}$ and $U_{ijs} = end_{T(s)} - st_{ij}$. In summary, we have

$$L_{ijs} = \begin{cases} start_{T(s)} - st_{ij} & \text{if } s = f^{T(s)} \\ start_{T(s)} & \text{if } s \neq f^{T(s)} \end{cases} \qquad \forall i, j \in \mathcal{I}, s \in \mathcal{S} \setminus \{1\} \tag{3a}$$

$$U_{ijs} = \begin{cases} start_{T(s)} & \text{if } s = f^{T(s)} \\ end_{T(s)} - st_{ij} & \text{if } s \neq f^{T(s)} \end{cases} \qquad \forall i, j \in \mathcal{I}, s \in \mathcal{S} \setminus \{1\} \tag{3b}$$

It is easy to modify both models so as not to allow setup crossover. For `(ST)`, we simply need to set $v_{ijt} = 0$ and $q_{ijt} = 0$. In this case, `(ST)` becomes equivalent to the model of Guimarães et al. (2014). For `(TF)`, it is sufficient to modify $L_{ijs} = U_{ijs} = start_{T(s)}$ for all $s = f^{T(s)}$ in (3), such that the first setup of a macroperiod starts at the very beginning of the macroperiod and there is no chance for setup crossover from the previous macroperiod.

In Guimarães et al. (2014), GLSP is classified as a product-oriented microperiod model. Therefore, following their classification scheme, `(ST)` and `(TF)` are also categorized as product-oriented microperiod models. In addition, Copil et al. (2017) classified GLSP as a generic model among various LSP models. Particularly, GLSP provides much flexibility in incorporating various extensions such as sequence-dependent or nontriangular setups as well as setup carryover, thanks to its two-level time structure. This is also true for `(ST)` and `(TF)` as they both are based on GLSP. Moreover, our models can further consider setup crossover which allows for even more flexibility than the basic GLSP.

3.3. Comparison of `(ST)` and `(TF)`

In this section, we compare the tightness of the LP relaxations of `(ST)` and `(TF)`. For comparison, let $\mathcal{P}^{\mathtt{ST}}$, $\mathcal{P}^{\mathtt{TF}}$ and $z^{\mathtt{ST}}_{\mathtt{LP}}$, $z^{\mathtt{TF}}_{\mathtt{LP}}$ be the sets of feasible solutions of the LP relaxations of `(ST)` and `(TF)` and the optimal objective values of their LP relaxation (LP bound, for short), respectively. For notational convenience, we use boldface to denote vectors and matrices; for example, $\mathbf{x} := (x_{is})_{i \in \mathcal{I}, s \in \mathcal{S}}$.

Proposition 3.1. $z^{\texttt{TF}}_{\texttt{LP}} \geq z^{\texttt{ST}}_{\texttt{LP}}$.

Proof. We show that for any given solution $(\bar{\mathbf{I}}, \bar{\mathbf{B}}, \bar{\mathbf{x}}, \bar{\mathbf{r}}, \bar{\mathbf{y}}, \bar{\mathbf{w}}, \bar{\mathbf{z}}) \in \mathcal{P}^{\texttt{TF}}$, the corresponding solution $(\mathbf{I}, \mathbf{B}, \mathbf{x}, \mathbf{y}, \mathbf{v}, \mathbf{q}, \mathbf{z}) \in \mathcal{P}^{\texttt{ST}}$ with the same objective value can be constructed. Firstly, by letting $(\mathbf{I}, \mathbf{B}, \mathbf{x}, \mathbf{y}, \mathbf{z}) = (\bar{\mathbf{I}}, \bar{\mathbf{B}}, \bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{z}})$, the constraints that are common in both models are satisfied. In addition, we set $v_{ijt} = start_t \bar{z}_{ijf^t} - \bar{w}_{ijf^t}$ and $q_{ijt} = \bar{z}_{ijf^t}$, for all $i, j \in \mathcal{I}$ and $t \in \mathcal{T}$. Then, constraints (1i) clearly hold. Moreover, constraints (1h) are satisfied by the bound constraints (2h). For a given $t \in \mathcal{T}$, the following result can be obtained by summing constraints (2c) for all $i \in \mathcal{I}$ and $s \in \mathcal{S}_t$:

$$\begin{aligned}\sum_{i\in\mathcal{I}}\sum_{s\in\mathcal{S}_t}\Big(\sum_{j\in\mathcal{I}} st_{ji}\bar{z}_{jis} + a_i\bar{x}_{is} + \bar{r}_{is}\Big) &= \sum_{i,j\in\mathcal{I}}\big(\bar{w}_{ijf^{t+1}} - \bar{w}_{ijf^t}\big)\\ &= \sum_{i,j\in\mathcal{I}}\Big(\big(start_{t+1}\bar{z}_{ijf^{t+1}} - v_{ijt+1}\big) - \big(start_t\bar{z}_{ijf^t} - v_{ijt}\big)\Big)\\ &= start_{t+1} - start_t - \sum_{i,j\in\mathcal{I}}\big(v_{ijt+1} - v_{ijt}\big) = K_t - \sum_{i,j\in\mathcal{I}}\big(v_{ijt+1} - v_{ijt}\big).\end{aligned}$$

The second equality holds by the relation between variables v_{ijt} and $\bar{w}_{ijf^t}$ constructed above, while the third holds as $\sum_{i,j\in\mathcal{I}} \bar{z}_{ijf^t} = 1$ for all $t \in \mathcal{T} \setminus \{1\}$ and $start_1 = 0$. Therefore, constraints (1c) of (ST) are also satisfied. Finally, it is clear that if constraints (2d) are satisfied, then constraints (1d) are satisfied. As a result, from any given solution in $\mathcal{P}^{\texttt{TF}}$, the corresponding solution that satisfies the set of constraints defining $\mathcal{P}^{\texttt{ST}}$ can be constructed. Moreover, because the objective functions are identical, their values are the same. □

The above proposition shows that the LP bound of (TF) is at least as strong as that of (ST), which means that $\mathcal{P}^{\texttt{TF}}$ is at least as tight as $\mathcal{P}^{\texttt{ST}}$. We then show that $\mathcal{P}^{\texttt{TF}}$ can be tighter than $\mathcal{P}^{\texttt{ST}}$. To this end, we derive a family of valid inequalities for (ST) using (TF). Let $\mathcal{X} := \{(i,s) : i \in \mathcal{I}, s \in \mathcal{S}\}$ be the set of pairs of an item and a microperiod that coincides with the nodes in Figure 1. Similarly, for a given macroperiod $t \in \mathcal{T}$, let $\mathcal{X}_t := \{(i,s) : i \in \mathcal{I}, s \in \mathcal{S}_t\}$. For ease of notation, we denote an arc $\big((i,s-1),(j,s)\big)$ as (i,j,s). For $X_1, X_2 \subseteq \mathcal{X}$, let $E(X_1, X_2)$ be the set of arcs (i,j,s) such that $(i,s-1) \in X_1$ and $(j,s) \in X_2$. In addition, for $X \subseteq \mathcal{X}$, let $E(X) := E(X,X)$, $\delta^+(X) := E(X, X^C)$, and $\delta^-(X) := E(X^C, X)$.

Proposition 3.2. *For a given subset of nodes $X \subseteq \mathcal{X}$, the inequality*

$$\begin{aligned}\sum_{(i,s)\in X} a_i x_{is} + \sum_{(j,i,s)\in E(X)} st_{ji}z_{jis} + \sum_{\substack{(j,i,s)\in\delta^-(X):\\ s\neq f^{T(s)}}} st_{ji}z_{jis} + \sum_{\substack{(i,k,s+1)\in\delta^+(X):\\ s\neq l^{T(s)}}} st_{ik}z_{iks+1}\\ \leq \sum_{(i,k,s+1)\in\delta^+(X)} end_{T(s)}z_{iks+1} - \sum_{(j,i,s)\in\delta^-(X)} start_{T(s)}z_{jis} \quad \forall X \subseteq \mathcal{X} \quad (4)\end{aligned}$$

is valid for (ST). *Moreover, if we restrict* X *to be a subset of* $\mathcal{X}_t$ *for a given* $t \in \mathcal{T}$, *inequality* (4) *is presented in a more concise form:*

$$\sum_{(i,s)\in X} a_i x_{is} + \sum_{(j,i,s)\in E(X)} st_{ji} z_{jis} + \sum_{\substack{(j,i,s)\in\delta^-(X):\\ s\neq f^t}} st_{ji} z_{jis} + \sum_{\substack{(i,j,s+1)\in\delta^+(X):\\ s\neq l^t}} st_{ik} z_{iks+1} \leq K_t \sum_{(j,i,s)\in\delta^-(X)} z_{jis} \qquad \forall X \subseteq \mathcal{X}_t, \quad (5)$$

Proof. See Appendix A. □

The meaning of inequality (5) is as follows: When $\sum_{(j,i,s)\in\delta^-(X)} z_{jis} = 0$, there are no incoming arcs to X; therefore, the nodes in X and the adjacent arcs cannot be visited. When $\sum_{(j,i,s)\in\delta^-(X)} z_{jis} \geq 1$, the right-hand side is greater than or equal to K_t. As the setup arcs that may cross over are not included in the summations of the left-hand side, the inequalities are satisfied owing to the capacity constraints. Next, we show that inequalities (4) are not implied in (ST), which means that $\mathcal{P}^{\mathtt{TF}}$ can be tighter than $\mathcal{P}^{\mathtt{ST}}$.

Proposition 3.3. *The inequalities* (4) *are not dominated by the constraints of* (ST).

Proof. As inequalities (5) belong to a family of valid inequalities (4), it is sufficient to show that inequalities (5) can cut off some fractional solutions of (ST). We consider the fractional solution of (ST) illustrated in Figure 2. This solution is constructed as follows: For a given $t \in \mathcal{T}$, let $f := f^t$ and $l := l^t$. Then, for a given $i \in \mathcal{I}$, let $y_{is} = 1$ for $s \in \mathcal{S}$ such that $s \leq f$ or $s \geq l$. Furthermore, let $z_{iif+1} = z_{iif+2} = \cdots = z_{iil} = \frac{1}{K_t}$ and $z_{ikf+1} = z_{kkf+2} = \cdots = z_{kkl-1} = z_{kil} = \frac{K_t-1}{K_t}$ for any $k \in \mathcal{I}$. We can easily observe that the constructed solution is feasible for $\mathcal{P}^{\mathtt{ST}}$.

By letting $X = \{(i, f+1), (i, f+2)\}$, $\sum_{(j,i,s)\in\delta^-(X)} z_{jis} = \frac{1}{K_t}$, and therefore, the right-hand side of the corresponding inequality (5) is equal to one. Meanwhile, due to the setup constraints

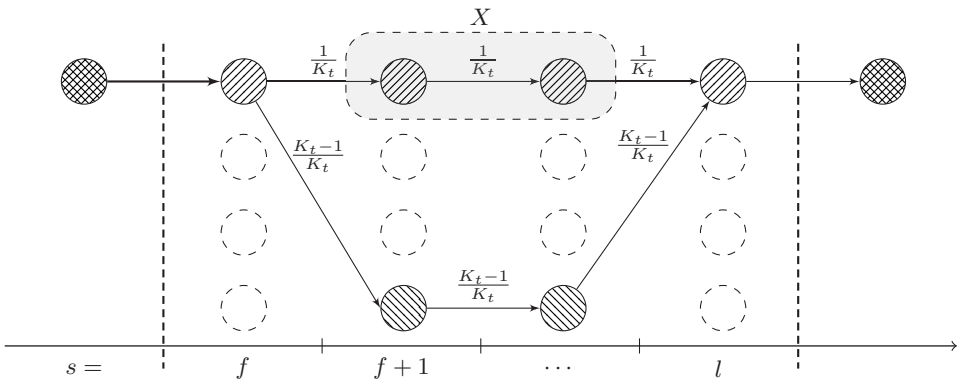

Figure 2: Illustration of the fractional solution of (ST)

$a_i x_{if+1} \leq K_t y_{if+1}$ and $a_i x_{if+2} \leq K_t y_{if+2}$, we can let $x_{if+1} = x_{if+2} = \frac{1}{a_i}$. In this case, the left-hand side of inequality (5) is at least greater than two. Therefore, inequality (5) is violated, which means that the fractional solution is cut off by it. □

With a slight abuse of notation, we define $\mathcal{P}^{\mathtt{ST}+(4)}$ be the feasible set of the LP relaxation of (ST) after adding the family of inequalities (4). In addition, for $\mathtt{F} \in \{\mathtt{ST}, \mathtt{ST}+(4), \mathtt{TF}\}$, let $\mathcal{Q}^{\mathtt{F}} := proj_{(\mathbf{I},\mathbf{B},\mathbf{x},\mathbf{y},\mathbf{z})}\mathcal{P}^{\mathtt{F}}$; that is, the projection of $\mathcal{P}^{\mathtt{F}}$ onto the space of variables $(\mathbf{I}, \mathbf{B}, \mathbf{x}, \mathbf{y}, \mathbf{z})$. The above results yield the following corollary, which implies that $z^{\mathtt{TF}}_{\mathtt{LP}}$ can be greater than $z^{\mathtt{ST}}_{\mathtt{LP}}$.

Corollary 3.4. $\mathcal{Q}^{\mathtt{TF}} \subseteq \mathcal{Q}^{\mathtt{ST}+(4)} \subsetneq \mathcal{Q}^{\mathtt{ST}}$.

3.4. Facility Location Reformulation

After firstly being introduced by Krarup & Bilde (1977), the facility location (FL) reformulation technique has been widely used for various LSP models. It is part of the folklore, so to speak, that this reformulation significantly improves the quality of the LP bound; see, for example, Pochet & Wolsey (2006). Our models also can be further tightened by the FL reformulation. The basic idea is to use variables representing the fraction of demand for a specific period satisfied by the production in another period. Let α_{ist} be the variable representing the fraction of the demand of item i in macroperiod t satisfied by the production in microperiod s. As backlogging is allowed, the demand in t can be served from any microperiod s. In addition, we use a dummy microperiod $S+1$ to represent the demand that is not satisfied during the entire planning horizon; for example, α_{iS+1t} represents the fraction of demand of item i in macroperiod t that is not satisfied. The following relations hold between the original variables $(\mathbf{x}, \mathbf{I}, \mathbf{B})$ and the new variable $\boldsymbol{\alpha}$.

$$x_{is} = \sum_{t \in \mathcal{T}} d_{it} \alpha_{ist} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S},$$

$$I_{it} = \sum_{l=t+1}^{T} \sum_{s=1}^{l^t} d_{il} \alpha_{isl} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T},$$

$$B_{it} = \sum_{l=1}^{t} \sum_{s=f^{t+1}}^{S+1} d_{il} \alpha_{isl} \qquad \forall i \in \mathcal{I}, t \in \mathcal{T}.$$

Using the above relations, (ST) and (TF) can be reformulated as (ST-FL) and (TF-FL), respectively. The complete models of (ST-FL) and (TF-FL) are given in Appendix B.

4. Approximate Dynamic Programming Algorithm

4.1. Dynamic Programming Reformulation

Before presenting the ADP algorithm, we first reformulate our problem as a DP problem. We regard microperiods $s = 1, \ldots, S$ as stages. The state of the system at stage s, denoted by R_s, is

defined as a sequence of items that are set up up to microperiod s. Specifically, let $R_s = (i_1, \ldots, i_s)$, where i_k denotes an index of the item that is set up in microperiod k. Thus, the state space at stage s is defined as $\mathcal{R}_s = \{(i_1, \ldots, i_s) |\, i_k \in \mathcal{I},\ \forall k = 1, \ldots, s\}$. The set of possible states at stage s, given a previous state $\bar{R}_{s-1} = (\bar{i}_1, \ldots, \bar{i}_{s-1}) \in \mathcal{R}_{s-1}$, is defined as

$$\mathcal{R}_s(\bar{R}_{s-1}) = \{(\bar{i}_1, \ldots, \bar{i}_{s-1}, i) |\, i \in \mathcal{I}\} = \{(\bar{R}_{s-1}, i) |\, i \in \mathcal{I}\}.$$

Let us define the value function $V_s(R_s)$ as the minimum total cost, except for the setup cost from 1 to s, under a given fixed sequence R_s. The value function of the last stage S for a given R_S, $V_S(R_S)$, can be easily computed by solving an LP, because the entire setup sequence is already determined. The problem is then to calculate the value function $V_0(R_0)$, where R_0 is a null sequence in which nothing is fixed. When a state R_{s-1} transitions to R_s, the corresponding state transition cost occurs. Specifically, if $R_{s-1} = (i_1, \ldots, i_{s-1})$ and $R_s = (i_1, \ldots, i_{s-1}, i_s)$, the transition cost $C_s(R_{s-1}, R_s)$ is the setup cost between i_{s-1} and i_s; that is, $sc_{i_{s-1}, i_s, T(s)}$. We assume that $C_1(R_0, \cdot) = 0$. With this definition, DP recursion can be expressed as

$$\begin{aligned} V_{s-1}(R_{s-1}) &= \min_{R_s \in \mathcal{R}_s(R_{s-1})} \left\{ C_s(R_{s-1}, R_s) + V_s(R_s) |\, R_{s-1} = (i_1, \ldots, i_{s-1}) \right\} \\ &= \min_{i \in \mathcal{I}} \left\{ sc_{i_{s-1}, i, T(s)} + V_s(i_1, \ldots, i_{s-1}, i) \right\}, \quad (6) \end{aligned}$$

for $s = 1, \ldots, S$. The last stage value function $V_S(\cdot)$ is easy to compute, as previously mentioned. Starting with $s = S$, $V_0(R_0)$ can be calculated using the backward recursion of equation (6). Subsequently, the optimal production sequence can be identified using

$$R_s^* = (R_{s-1}^*, i_s^*) = (i_1^*, \ldots, i_s^*) \text{ where } i_s^* = \arg\min_{i \in \mathcal{I}} \{ sc_{i_{s-1}^*, i, T(s)} + V_s(i_1^*, \ldots, i_{s-1}^*, i) \}$$

for $s = 1, \ldots, S$ and $R_0^* = R_0$. However, as there is an exponential number of $\mathcal{O}(I^S)$ states, the number of states easily explodes when the number of items and periods increases. Therefore, it is not practical to solve DP recursion exactly. In the following section, we present an ADP algorithm to avoid the issue of the curse of dimensionality.

4.2. Approximate Dynamic Programming Algorithm

In our ADP algorithm, we use the approximated value function $\hat{V}_s(R_s)$, which approximates $C_s(R_{s-1}, R_s) + V_s(R_s)$. Then, recursion (6) is modified as

$$\tilde{V}_{s-1}(R_{s-1}) = \min_{R_s \in \mathcal{R}_s(R_{s-1})} \left\{ \hat{V}_s(R_s) |\, R_{s-1} \right\} \quad (7)$$

where $\tilde{V}_{s-1}(R_{s-1})$ is an estimate of the value of being at R_{s-1}. The approximated value function $\hat{V}_s(R_s)$ is constructed using both the lower bound $LB(R_s)$ and the upper bound $UB(R_s)$ of the

Algorithm 1: LPNF($\mathcal{P}$, γ^{up}, γ^{down}, δ, $\mathcal{Y}_0$, $\mathcal{Y}_1$)

```
1  flag ← true;                                                          // set the flag
2  (obj, y^LP) ← solveLP(P, Y0, Y1)          // get obj. value and sol. value of variable y
3  repeat
4      if y^LP all_binary then                          // if all y variables are binary,
5          (UB^LPNF, y^LPNF) ← (obj, y^LP);             // return UB and feasible sol.
6          break;                                                        // escape
7      end
8      forall (i, s) ∈ (I × S) \ {Y0 ∪ Y1} do
9          if y_is^LP > γ^up then                                        // fix to one
10             Y1 ← Y1 ∪ {(i, s)};
11             flag ← false;
12         else if y_is^LP < γ^down then                                 // fix to zero
13             Y0 ← Y0 ∪ {(i, s)};
14             flag ← false;
15         end
16     end
17     if flag then                  // if no variables are fixed in current iteration
18         pick δ least fractional variables and put them in Y0 or Y1 ;  // fix δ variables
19     end
20     flag ← true;                                                      // reset the flag
21     (obj, y^LP) ← solveLP(P, Y0, Y1);
22 until false;
23 return (UB^LPNF, y^LPNF)
```

true value of $C_s(R_{s-1}, R_s) + V_s(R_s)$. There are many different ways to obtain these bounds. One can expect that if tighter $LB(R_s)$ and $UB(R_s)$ are used, the approximation will be more accurate and the solution quality will be improved. In addition, as these bounds need to be obtained repetitively in the ADP algorithm, it is important to do so in a short time. Considering this trade-off, acquiring good bounds in a short time is a key aspect of the ADP algorithm. In this regard, to obtain $LB(R_s)$, we solve the LP relaxation of the problem with the partially fixed production sequence $R_s = (i_1, \ldots, i_s)$; that is, $y_{i_k k} = 1$ for all $1 \leq k \leq s$. To obtain $UB(R_s)$ in the ADP algorithm, we use a simple LP-based naive fixing (LPNF) heuristic algorithm as a subroutine.

The LPNF algorithm for an instance $\mathcal{P}$ is described in Algorithm 1. In the algorithm, LP relaxation is recursively solved (lines 2, 21) with the subroutine $solveLP(\mathcal{P}, \mathcal{Y}_0, \mathcal{Y}_1)$ which provides the solution and the objective value of the LP relaxation of $\mathcal{P}$. The input parameters $\mathcal{Y}_0$ and $\mathcal{Y}_1$ are the index sets of variable $\mathbf{y}$, which are fixed to zero and one, respectively. After obtaining the fractional solution, variables whose values are greater than an upper threshold γ^{up} ($0.5 \leq \gamma^{up} < 1$) are fixed to one, while variables whose values are less than a lower threshold γ^{down} ($0 < \gamma^{down} < 0.5$) are fixed to zero (lines 8–16). This procedure is repeated until all binary variables are fixed to zero or one (lines 4–7). We only consider the binary condition of variable $\mathbf{y}$, because if it is fixed to binary values, the variable $\mathbf{z}$ is also restricted to binary values.

If all values of the nonfixed binary variables are neither below the lower threshold nor above the upper threshold, during the iterations, the algorithm can run permanently. To avoid this, if

Algorithm 2: ADP$(\mathcal{P}, \varepsilon, \lambda, \gamma^{up}, \gamma^{down}, \delta)$

1 $\mathcal{Y}_1 \leftarrow \emptyset$, $\mathcal{Y}_0 \leftarrow \emptyset$, $s \leftarrow 1$, $UB^{ADP} \leftarrow \infty$; `// initialization`
2 $(obj, \mathbf{y}^{LP}) \leftarrow solveLP(\mathcal{P}, \mathcal{Y}_0, \mathcal{Y}_1)$; `// solve LP relaxation`
3 $\mathcal{CAND}(s) \leftarrow getCand(\mathbf{y}^{LP}, \lambda, s)$; `// select candidates to be evaluated at stage` s
4 **do**
5 | **forall** $c \in \mathcal{CAND}(s)$ **do** `// evaluate each candidate state`
6 | | $\big(LB(c), \mathbf{y}^{LP}(c)\big) \leftarrow solveLP(\mathcal{P}, \mathcal{Y}_0, \mathcal{Y}_1 \cup \{c\})$;
7 | | $\big(UB(c), \mathbf{y}^{LPNF}(c)\big) \leftarrow LPNF(\mathcal{P}, \gamma^{up}, \gamma^{down}, \delta, \mathcal{Y}_0, \mathcal{Y}_1 \cup \{c\})$;
8 | | **if** $UB(c) < UB^{ADP}$ **then** `// if better solution found`
9 | | | $(UB^{ADP}, \mathbf{y}^{ADP}) \leftarrow (UB(c), \mathbf{y}^{LPNF}(c))$
10 | | **end**
11 | | $\hat{V}(c) \leftarrow (1-\varepsilon) \cdot LB(c) + \varepsilon \cdot UB(c)$;
12 | **end**
13 | $c^* \leftarrow \underset{c \in \mathcal{CAND}(s)}{\arg\min}\ \hat{V}(c)$;
14 | $\mathcal{Y}_1 \leftarrow \mathcal{Y}_1 \cup \{c^*\}$;
15 | $\mathcal{CAND}(s+1) \leftarrow getCand(\mathbf{y}^{LP}(c^*), \lambda, s+1)$;
16 | $s \leftarrow s+1$;
17 **while** $s < S$;
18 $(obj, \mathbf{y}^{LP}) \leftarrow solveLP(\mathcal{P}, \mathcal{Y}_0, \mathcal{Y}_1)$;
19 **if** $obj < UB^{ADP}$ **then**
20 | $(UB^{ADP}, \mathbf{y}^{ADP}) \leftarrow (obj, \mathbf{y}^{LP})$
21 **end**
22 **return** $(UB^{ADP}, \mathbf{y}^{ADP})$

there are no variables to be fixed in an iteration, the least fractional variables are forcibly fixed to zero or one, whichever is closer (lines 17–19). The number of variables that can be forcibly fixed in one iteration is indicated by an integer parameter δ. Note that the LPNF algorithm for a given instance $\mathcal{P}$ can use any of the previously presented models: `(ST)`, `(TF)`, `(ST-FL)`, or `(TF-FL)`.

With $LB(R_s)$ and $UB(R_s)$, calculated by the LP relaxation and the LPNF algorithm, respectively, we evaluate $\hat{V}_s(R_s)$ using the equation

$$\hat{V}_s(R_s) = (1-\varepsilon) \cdot LB(R_s) + \varepsilon \cdot UB(R_s)$$

where parameter ε indicates the ratio of the upper bound when estimating the value of state ($0 \leq \varepsilon \leq 1$). If we set $\varepsilon = 1$ ($\varepsilon = 0$), the value is approximated using only the upper bound (lower bound). Using this approximation, the production sequence can be identified using

$$\hat{R}_s = (\hat{R}_{s-1}, \hat{i}_s) = (\hat{i}_1, \ldots, \hat{i}_s) \text{ where } \hat{i}_s = \underset{i \in \mathcal{I}}{\arg\min}\{\hat{V}_s(\hat{i}_1, \ldots, \hat{i}_{s-1}, i)\}$$

for $s = 1, \ldots, S$ and $\hat{R}_0 = R_0$. Moreover, given a state, it is possible to consider only some promising states among all possible next states so as to avoid wasting computational time on evaluating prospectless states. The set of promising candidate states, denoted by $\mathcal{CAND}$, is chosen as those with a large LP fractional solution value. An integer parameter $1 \leq \lambda \leq I$ is used to

indicate the number of states to be evaluated. By setting $\lambda = I$, every possible future state is evaluated. If $\lambda = 1$, only one future state, which is to say, that with the largest LP fractional solution value, is evaluated. The overall ADP algorithm for an instance $\mathcal{P}$ is given in Algorithm 2. Starting from stage 1, at each stage, the algorithm fixes the state with the best approximated value until it reaches the last stage and returns the solution $\mathbf{y}^{ADP}$ and its objective value UB^{ADP}. While executing the algorithm, UB^{ADP} is updated whenever a better solution is obtained. In this algorithm, a subroutine $getCand(\mathbf{y}^{LP}, \lambda, s)$, which selects candidate states to be evaluated at stage s, is used. From the fractional solution $\mathbf{y}^{LP}$, this subroutine compares the y_{is}^{LP} values for $i \in \mathcal{I}$ and returns a set of indices of λ largest fractional values: $\mathcal{CAND}(s) = \{(i^1, s), \ldots, (i^\lambda, s)\}$.

5. Computational Experiments

We conduct two computational experiments. The first aims to investigate the performance of the presented GLSP-based models and test the solution algorithms using a set of instances similar to a specific real-world manufacturing environment. In the second experiment, the time-flow model and the solution algorithm are further compared with a recent big bucket model using a set of instances from the literature.

5.1. Comparison of GLSP-based Models

5.1.1. Test Instances

The first set of instances is generated considering the characteristics of the manufacturing environment introduced in Lee & Lee (2020). In this manufacturing process, the setup takes a significant amount of time, and thus, it should not occur frequently in each macroperiod. In this regard, we set $|\mathcal{S}_t| = 3$ for all $t \in \mathcal{T}$. The length of each macroperiod is set to 100 (i.e., $K_t = K = 100$), and the unit production time a_i is set to 1 for all instances. The dimension of an instance is defined by the number of items (I) and the number of macroperiods (T). We use five dimensions: $(5, 10)$, $(5, 15)$, $(10, 10)$, $(10, 15)$, and $(15, 15)$.

In order to generate demand data similar to the demand patterns of this manufacturing environment, we use two parameters ρ and f, which denote the production capacity utilization level and the demand frequency, respectively. Specifically, $\rho = \frac{\sum_{i\in\mathcal{I},t\in\mathcal{T}} d_{it}}{K\cdot T}$ and $f = \frac{\sum_{i\in\mathcal{I},t\in\mathcal{T}} \mathbb{I}_{\{d_{it}>0\}}}{I\cdot T}$ where $\mathbb{I}_{\{d_{it}>0\}} = 1$ if nonzero demand occurs for item i in macroperiod t. From these parameters, the average amount of nonzero demand can be determined; that is, $d_{avg} = \frac{K\cdot\rho}{I\cdot f}$. Among the elements of the set $\mathcal{I} \times \mathcal{T}$, we randomly choose $f \cdot I \cdot T$ demand points while ensuring that at least one item has nonzero demand in the last macroperiod and that there is no item with zero demand during the entire planning horizon. For each selected demand point, the amount of demand is generated from a discrete uniform distribution $[0.8 \cdot d_{avg}, 1.2 \cdot d_{avg}]$. We use the parameters $\rho \in \{0.8, 1\}$, $f \in \{1/3, 1/2\}$.

To determine the influence of the length of the setup times, we use three different classes: S, M, and L, whose (st^{min}, st^{max}) are $(0.05, 0.15)$, $(0.1, 0.3)$, and $(0.4, 0.6)$, respectively. Then, the setup

times are generated from a discrete uniform distribution $DU[st^{min} \cdot K, st^{max} \cdot K]$. The inventory holding cost hc_{it} is set to 1, whereas the backlogging cost bc_{it} is generated from $DU[1, 5]$. The setup costs are set to be the same as the setup times; that is, $sc_{ijt} = st_{ij}$.

There are $5 \times 3 \times 4 = 60$ different combinations of factors, and we generate five instances for each combination, such that 300 instances are presented in total. The experimental results are averaged over these five instances of a particular combination. All of our experiments were conducted on an Intel Core 3.10 GHz PC with 16 GB RAM under Windows 10 Pro. The proposed algorithms were implemented in C++. FICO Xpress 8.9 with its default parameter settings was used as the LP/MIP solver. We set the time limit for solving an MIP problem to 600 s.

5.1.2. LP Bound

The results of the tightness of the LP bound (LPB) are presented in Table 2. They are summarized for each factor, as indicated in the first column. The *LP gap* is computed as $\frac{(MIP\ Best\ Sol)-(LPB)}{MIP\ Best\ Sol} \times 100\%$, where *MIP Best Sol* is the best solution among those obtained by solving the four models using the MIP solver. The column *Gap Closed* for each model represents the ratio of the closed gap with respect to the LPB of `ST`; that is, $\frac{(LPB \text{ of the model})-(LPB \text{ of } \texttt{ST})}{(MIP\ Best\ Sol)-(LPB \text{ of } \texttt{ST})} \times 100\%$.

As shown in Table 2, the `FL` reformulation has a significant impact on tightening the LPB for both `(ST)` and `(TF)`. The average LP gap of the models with the `FL` reformulation is about 65%, whereas that of the models without the `FL` reformulation is above 80%. From these results, we can see that, similarly to other LSP models, our models can benefit significantly from `FL` reformulation.

In addition, there is an obvious improvement in LPB if `(TF)` is used instead of `(ST)`. The *LP gap*

Table 2: Comparison of the LP bound of the models

Factors	*LP Gap* (%)				*Gap Closed* (%)		
	ST	TF	ST-FL	TF-FL	TF	ST-FL	TF-FL
Dim							
5×10	74.6	63.1	70.6	62.2	16.6	5.2	17.7
5×15	73.2	62.4	69.4	61.7	16.2	5.0	17.0
10×10	91.6	84.4	65.9	63.8	8.0	27.9	30.3
10×15	92.1	85.3	70.0	67.9	7.6	23.7	26.1
15×15	97.2	93.2	65.4	63.9	4.1	32.7	34.2
Setup Time							
S	77.4	70.3	51.1	47.3	11.0	29.0	35.7
M	86.1	77.9	69.8	65.2	10.7	17.4	23.8
L	93.7	84.8	83.9	79.1	9.8	10.2	15.6
Demand							
$(0.8, 1/3)$	85.8	77.7	67.7	63.3	10.2	19.7	25.6
$(0.8, 1/2)$	95.2	90.8	67.0	65.7	4.9	29.0	30.4
$(1, 1/3)$	76.8	64.9	68.2	60.7	17.3	10.0	22.0
$(1, 1/2)$	85.2	77.4	70.1	65.9	9.7	16.8	22.1
Total	85.7	77.7	68.3	63.9	10.5	18.9	25.1

is reduced by approximately 8% on average. This also holds between (ST-FL) and (TF-FL). The *LPB* provided by (TF-FL) is tighter than that by (ST-FL). In summary, (TF-FL) is the tightest model, and approximately 25% of the gap is closed compared to (ST). The following relationship is shown between the tightness of the models: (TF-FL) > (ST-FL) > (TF) > (ST).

5.1.3. Computational Performance with MIP Solver

Now, we compare the solution quality and computational burden when the MIP solver is used. The results are listed in Table 3. Column *Gap* shows the final gap between the best solution and the best lower bound found by the solver within the time limit; for example, a zero gap indicates that the optimal solution is found. The ratio of the instances that found the optimal solution among all corresponding instances is represented in column *Opt*. The next columns *#Node* and *Time* represent the average number of nodes visited while running the branch-and-bound algorithm and the average computation time, respectively.

The main observation is that (TF) and (TF-FL) are more successful in solving problem instances than are (ST) and (ST-FL). The average gap of (ST) and (ST-FL) is approximately 35%, whereas that of (TF) and (TF-FL) is approximately 27%. Meanwhile, even if the FL reformulation is effective in reducing the LP gap, it does not seem to help much in improving the MIP solvability. With the reformulation, the final gap increases, and the number of optimal solutions found diminishes. This is due to the fact that the disadvantage of the increased model size is greater than the gain from the tighter LP bound. In conclusion, (TF) shows the best performance.

The number of nodes visited by (TF) and (TF-FL) to close the gap is much smaller than those by (ST) and (ST-FL). Furthermore, they require less computation time. In summary, the proposed idea of the time-flow model seems quite useful. For the largest instances of dimensions $(15, 15)$, however, the performance is not satisfactory, because the average gap is above 50%. Therefore, we applied our solution approaches to those instances.

5.1.4. Performance of LPNF Algorithm

We first test the performance of the standalone LPNF algorithm. Specifically, we investigate the effects of the tightness of the base model and the different parameter settings. We compare the different parameters $\gamma^{up} \in \{0.7, 0.9\}$ and $\delta \in \{1, 2, 3, 4, 5\}$, while γ^{down} is fixed to zero. Table 4 presents the relative solution values and computation times where the relative solution value is defined as $\frac{\text{LPNF solution value}}{\textit{MIP Best Sol}} \times 100\%$.

As shown in Table 4, there are significant differences in the algorithms' performance with the different models. The quality of the solution obtained using (TF-FL) is much better than that obtained using the others. The average deviation from the *MIP Best Sol* is approximately 257% with (ST), whereas it is about only 24% with (TF-FL). These results show that the tightness of the base model greatly affects the solution quality, which is natural, as the algorithm is highly dependent on the quality of the LP bound. Among all of the models, (TF-FL), the tightest,

Table 3: Comparison of the computational performance of the models

Factors	*Gap* (%)				*Opt* (%)				*# Node*				*Time* (s)			
	ST	TF	ST-FL	TF-FL	ST	TF	ST-FL	TF-FL	ST	TF	ST-FL	TF-FL	ST	TF	ST-FL	TF-FL
Dim																
5×10	1.0	0.1	2.4	0.0	96.7	98.3	83.3	100.0	306301	9609	123924	7899	103.7	37.0	179.2	59.6
5×15	18.8	4.4	23.7	8.1	35.0	71.7	28.3	56.7	814890	50286	150243	22390	450.4	305.1	504.3	391.4
10×10	31.5	25.5	35.7	35.6	11.7	18.3	10.0	5.0	209644	5562	67427	2456	555.2	539.0	563.7	579.0
10×15	50.9	40.5	57.4	49.9	3.3	6.7	1.7	1.7	113881	2186	12330	708	589.4	572.3	593.9	598.2
15×15	61.1	53.6	64.4	58.9	0.0	0.0	0.0	0.0	19461	37	2480	5	600.0	600.0	600.0	600.0
Setup Time																
S	18.4	11.4	21.1	17.2	41.0	54.0	39.0	28.0	212255	8141	52716	5549	396.3	340.6	420.1	411.1
M	35.3	25.4	39.8	31.5	24.0	32.0	18.0	21.0	327046	16186	81573	7321	483.9	440.5	510.6	464.7
L	44.3	37.7	49.3	42.8	23.0	31.0	17.0	29.0	339204	16281	79553	7204	498.9	450.9	533.9	461.2
Demand																
$(0.8, 1/3)$	30.2	21.0	32.4	27.4	36.0	46.7	33.3	26.7	257561	9636	72453	6037	423.6	368.5	446.3	418.2
$(0.8, 1/2)$	33.8	24.6	36.4	29.9	28.0	38.7	21.3	52.0	298199	17258	69246	7078	475.1	429.7	510.3	476.1
$(1, 1/3)$	30.4	24.8	36.3	29.6	29.3	38.7	26.7	38.7	271736	9605	77482	5814	444.9	402.1	478.3	413.9
$(1, 1/2)$	36.3	28.9	41.8	35.1	24.0	32.0	17.3	40.0	343845	17645	65942	7836	495.2	442.5	518.0	474.4
Total	32.7	24.8	36.7	30.5	29.3	39.0	24.7	32.7	292835	13536	71281	6691	459.7	410.7	488.2	445.6

Table 4: Test results for the LPNF algorithm

γ^{up}	δ	*Relative Solution Value* (%)				*Time* (s)			
		ST	TF	ST-FL	TF-FL	ST	TF	ST-FL	TF-FL
0.7	1	263.0	178.3	120.3	106.5	3.5	6.9	3.9	6.8
	2	276.6	175.5	138.7	116.6	2.5	4.8	2.6	4.7
	3	496.2	179.7	245.2	126.7	2.4	3.9	2.1	4.0
	4	423.7	182.7	201.4	132.8	1.9	3.3	1.8	3.4
	5	330.9	189.8	202.8	133.8	1.7	3.0	1.6	3.1
0.9	1	253.2	170.2	123.1	107.2	3.6	6.9	3.8	6.7
	2	280.3	174.4	140.1	116.1	2.5	4.7	2.6	4.7
	3	488.5	180.6	240.6	129.9	2.5	3.8	2.1	3.9
	4	432.3	185.6	200.9	133.7	2.5	3.3	1.8	3.4
	5	328.4	186.9	201.4	133.0	2.8	2.9	1.6	3.1
Average		357.3	180.4	181.5	123.6	2.6	4.4	2.4	4.4

shows the best solution quality. The computation time for `(TF-FL)` is approximately twice that of `(ST-FL)`, but is short enough to be used for the ADP algorithm. The parameter γ^{up} does not affect either the solution quality or time. The larger δ leads to shorter computation time at the expense of slightly deteriorated solution quality, because more variables can be fixed in each iteration.

5.1.5. Performance of ADP Algorithm

For the ADP algorithm, we fix the threshold parameters $(\gamma^{down}, \gamma^{up})$ to $(0, 0.9)$ and use `TF-FL` as the base model, which shows good performance with the LPNF algorithm. We use fixing parameters $\delta \in \{1, 3, 5\}$, candidates parameters $\lambda \in \{2, 3, 5\}$, and weighting parameters $\epsilon \in \{0, 0.2, 0.5, 0.8, 1\}$. Note that when $\epsilon = 0$, the parameter δ has no meaning because the LPNF algorithm is not used.

Table 5 reports the relative solution values, the ratio of instances whose ADP solution is better than the *MIP Best Sol*, and the computation time. The relative solution values and computation time are also illustrated in Figures 3 and 4, respectively. In Figure 4, the computation time for the ADP algorithms with ϵ greater than zero are averaged and grouped together because there are no meaningful differences between them.

Overall, the performance of the ADP algorithm seems satisfactory. As shown in Table 5, the largest average deviation from *MIP Best Sol* is only about 4% when the LP bound is not used ($\epsilon = 1$). When both the LP bound and the LPNF algorithm are used, the ADP algorithms succeed in finding even better solutions under most of the parameter settings. For instance, when $(\lambda, \delta, \epsilon) = (5, 1, 0.2)$ is used, the solution quality improves by more than 10% compared with *MIP Best Sol*. Under this setting, the ADP algorithm obtains better solutions for 90% of the instances.

Regarding the parameters, $\delta = 1$ and $\epsilon = 0.2$ show, as indicated in Figure 3, the best results with respect to the relative solution value. Smaller δ leads to better solutions at the expense of increased computation times. The average computation time is also proportional to the value of

Table 5: Test results for the ADP algorithm

λ	δ	ϵ	*Relative Solution Value* (%)	*Better Solution* (%)	*Time* (s)
2	-	0	93.54	80.00	66.5
	1	0.2	92.46	81.67	263.6
		0.5	94.81	76.67	259.4
		0.8	95.78	76.67	251.1
		1	95.84	75.00	272.0
	3	0.2	96.28	66.67	174.8
		0.5	98.94	61.67	170.1
		0.8	99.03	61.67	164.8
		1	99.38	56.67	183.6
	5	0.2	98.01	65.00	152.6
		0.5	101.08	55.00	163.2
		0.8	101.97	50.00	142.1
		1	102.45	48.33	170.7
	Average		97.66	65.77	187.3
3	-	0	91.65	83.33	99.8
	1	0.2	90.59	85.00	420.1
		0.5	95.17	75.00	384.2
		0.8	96.05	70.00	366.3
		1	96.09	70.00	407.4
	3	0.2	96.89	66.67	260.5
		0.5	100.15	56.67	249.1
		0.8	101.12	55.00	236.1
		1	101.50	55.00	270.1
	5	0.2	96.92	63.33	224.3
		0.5	100.92	60.00	237.1
		0.8	103.43	40.00	206.4
		1	103.68	48.33	252.2
	Average		98.01	64.49	278.0
5	-	0	89.82	91.67	158.2
	1	0.2	88.98	90.00	555.0
		0.5	94.72	71.67	569.3
		0.8	95.59	71.67	549.6
		1	96.20	70.00	590.8
	3	0.2	95.68	70.00	431.2
		0.5	100.79	60.00	407.6
		0.8	101.86	55.00	371.4
		1	101.86	55.00	474.2
	5	0.2	97.86	63.33	373.8
		0.5	102.62	55.00	373.1
		0.8	103.48	53.33	318.0
		1	103.93	51.67	412.3
	Average		97.95	66.03	429.6

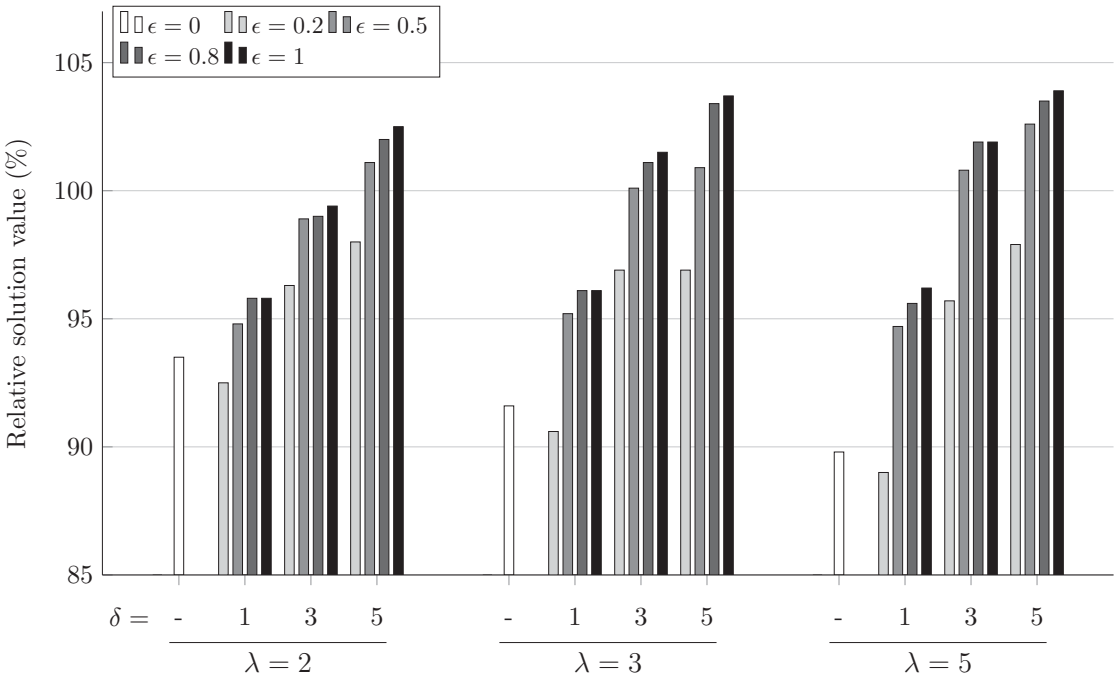

Figure 3: Test results for the ADP algorithm: Solution quality

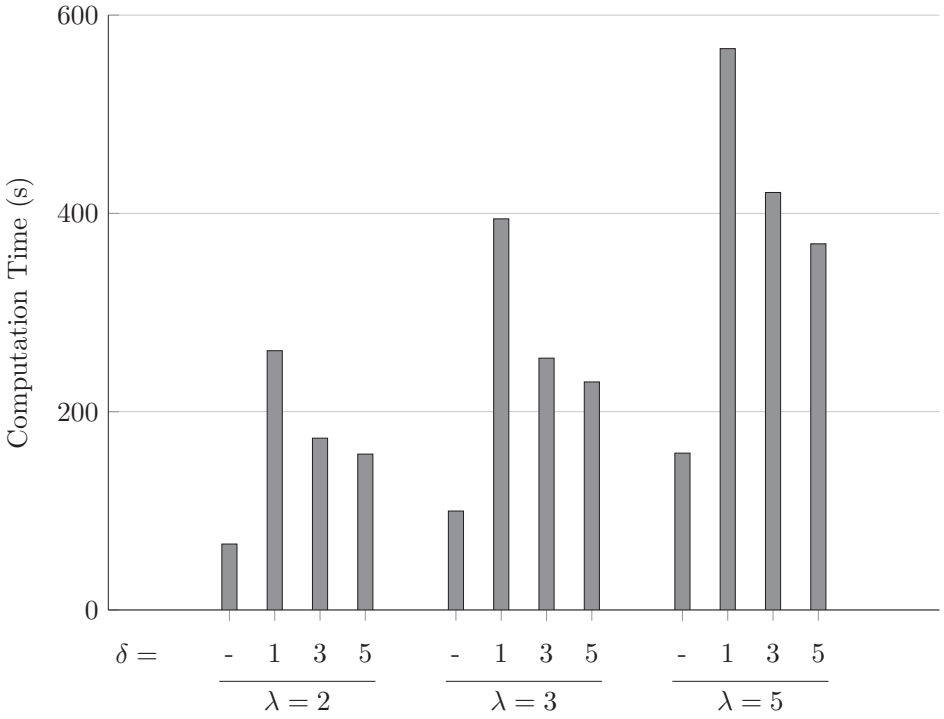

Figure 4: Test results for the ADP algorithm: Computation time

λ. Moreover, we observe that a larger λ does not always lead to better solution quality. For ϵ values with relatively good performance, a larger λ value helps improve the solution quality. On the contrary, for ϵ values with poor performance, evaluating additional states makes it even poorer.

5.2. Comparison with Big Bucket Model

In our second experiment, we further investigate the performance of the `(TF)` and ADP algorithm relative to a big bucket model recently proposed by Mahdieh et al. (2018). This model, denoted as `(MCB)`, uses a multi-commodity flow formulation to capture the sequence of production within each bucket. As far as we know, `(MCB)` is the latest LSP model that can incorporate the sequence-dependent setup, setup crossover, and setup carryover. Note that the models presented in Guimarães et al. (2014) cannot incorporate them all simultaneously. Moreover, `(MCB)` is an improved version of the models previously presented in Menezes et al. (2011) and Clark et al. (2014).

For the comparison, we create a set of instances according to the instance generation scheme in Almada-Lobo et al. (2007), which is frequently used in the literature (e.g. James & Almada-Lobo, 2011; Guimarães et al., 2014). The instance type is defined by the combination of problem dimensions ($I \times T$), capacity utilization (Cut), and setup cost factor (θ). See Almada-Lobo et al. (2007) for further details. We additionally introduce one more parameter to control the average length of the setup time *(Setup Time)*, as in the instances of the first experiment. We use the

Table 6: Comparison of the models and the solution algorithms

Factors	*Relative Solution Value* (%)					*Time* (s)			
	`TF`	`MCB`	$\texttt{ADP}_{\texttt{LU}}$	$\texttt{ADP}_{\texttt{L}}$	`LPNF`	`TF/MCB`	$\texttt{ADP}_{\texttt{LU}}$	$\texttt{ADP}_{\texttt{L}}$	`LPNF`
Dim									
10×10	100.0	96.61	101.88	105.26	112.15	1800	120.8	24.6	2.1
10×15	100.0	98.40	98.57	103.00	109.60	1800	403.8	92.3	6.7
15×10	100.0	96.34	95.94	97.44	105.52	1800	256.0	72.5	4.8
15×15	100.0	98.04	91.43	91.92	100.29	1800	537.7	242.0	14.2
Cut									
0.6	100.0	95.78	96.43	99.24	109.63	1800	337.4	97.5	6.4
0.8	100.0	97.68	96.28	98.91	105.67	1800	332.4	113.1	7.0
1	100.0	98.59	98.14	100.07	105.37	1800	318.9	113.0	7.4
θ									
50	100.0	97.24	96.40	98.49	106.73	1800	333.4	112.4	7.1
100	100.0	97.45	97.51	100.32	107.05	1800	325.7	103.4	6.8
Setup Time									
S	100.0	91.82	93.60	97.60	104.64	1800	658.7	227.3	13.6
M	100.0	98.40	98.15	99.89	107.34	1800	247.4	69.3	5.0
L	100.0	101.83	99.11	100.74	108.69	1800	82.6	27.0	2.3
Total	100.0	97.35	96.95	99.41	106.89	1800	329.6	107.9	7.0

following parameters: $I \in \{10, 15\}$, $T \in \{10, 15\}$, $Cut \in \{0.6, 0.8, 1\}$, $\theta \in \{50, 100\}$, and *Setup Time* $\in \{S, M, L\}$. The (st^{min}, st^{max}) of classes S, M, and L are $(0.2, 0.4)$, $(0.4, 0.6)$, and $(0.6, 0.8)$, respectively. For each combination, we generate five instances, resulting in a total of 360 instances. For these instances, we set the time limit for solving an MIP problem to 1800 s, and for the ADP algorithm, 900 s.

We compare the results obtained by solving `(TF)` and `(MCB)` with the MIP solver and those obtained by our ADP and LPNF algorithms. For the algorithms, we use `(TF-FL)` as a base model and set $(\gamma^{down}, \gamma^{up}, \delta, \lambda) = (0, 0.9, 3, 3)$. The corresponding results are presented in Table 6. We let the solution values of `(TF)` as 100 and indicate the relative solution quality of other approaches. The subscript of the ADP algorithm indicates the types of bounds used; that is, $\texttt{ADP}_{\texttt{LU}}$ uses both the lower and upper bounds (using $\epsilon = 0.2$), while $\texttt{ADP}_{\texttt{L}}$ uses only the lower bound ($\epsilon = 0$).

In the comparison of the two models, `(TF)` performs worse than `(MCB)` on average, though it is better in some instances. This can be explained by the fact that, due to the microperiods, the size of `(TF)` is larger than that of `(MCB)`, which makes it harder to solve. The performance of `(TF)` gets better as the setup time increases, which appears to be because the maximum number of items that can be produced within a macroperiod decreases, resulting in fewer microperiods. This result indicates that `(TF)` can be beneficial when the number of setups that can be conducted within a macroperiod is limited. Although `(MCB)` shows relatively better performance than `(TF)`, it is not successful in solving instances within the time limit.

The solution provided by $\texttt{ADP}_{\texttt{LU}}$ is 3% better than that of `(TF)` on average. Moreover, it is even better than the solution of `(MCB)`, though the difference is quite small. Similar to the results of the first experiment, $\texttt{ADP}_{\texttt{LU}}$ performs better than $\texttt{ADP}_{\texttt{L}}$ with longer computation time. Notably, the relative performance of both $\texttt{ADP}_{\texttt{LU}}$ and $\texttt{ADP}_{\texttt{L}}$ gets better as the size of instances increases. This indicates that the ADP algorithms can be more beneficial for larger instances. The computation time required for the ADP algorithms is much shorter than 1800 s.

However, the performance of the ADP algorithms relies on the choice of the parameters. Especially, in contrast to other parameters whose effects are quite predictable, choosing the value of weighting parameter ϵ may require several trials. In spite of the additional effort for choosing appropriate parameters, considering its advantages in terms of computation time, the ADP algorithm can be one viable option in solving LSPs with sequence-dependent setups.

6. Conclusion

In this paper, we introduce new integer optimization models for the LSP with sequence-dependent setups that can consider setup crossover and carryover. It is shown that the newly proposed time-flow models `(TF)` and `(TF-FL)` have certain benefits compared with the standard GLSP-based models in terms of the tightness of the LP bound and solvability with the MIP solver. Moreover, it is demonstrated that the tightness of the model not only affects the solvability with

the MIP solver, but also has a significant impact on the performance of the proposed algorithm. In the first experiment, our ADP algorithm shows some benefits over the MIP solver; that is, it can find a better solution within a shorter computation time. In addition, in the second experiment, the ADP algorithms show competitive performance in comparison with a state-of-the-art big bucket model in the literature.

The results of this study can be extended to big bucket models that consider sequence-dependent setups. When the setup occurs in a sequence-dependent manner, the big bucket models should consider the sequence of production lots within each bucket. This has been done by using cutset-type subtour elimination constraints, Miller-Tucker-Zemlin formulations, or single/multi-commodity-flow-based formulations (see Guimarães et al., 2014). In addition to these approaches, our time-flow model can also be used to present the production sequences within each bucket. Comparing the performances of these various methods seems to be an interesting future research direction. In addition, a family of valid inequalities similar to the proposed ones can be derived for the big bucket models. These inequalities can be used as cutting planes in a branch-and-cut algorithm, which represents another possible extension.

As another future research direction, there are some possible improvements to the proposed ADP algorithm. For instance, different schemes could be used to obtain primal and dual bound values instead of the LPNF algorithm and LP relaxation, respectively. In fact, even whether it is a valid bound or not is unimportant when approximating the values. It is sufficient if the true value can be approximated appropriately. Further, one can newly define the state in each stage, such as the inventory position of the items or the cumulative production amounts. With these states, the value function can be approximated using various techniques such as piecewise linear function fitting, regression, or machine-learning-based techniques.

Finally, the algorithm can be extended for application to cases in which uncertainty is present in data (e.g., demand or setup time). In such cases, the expected value of each state should be approximated, which can be achieved by various methods such as sampling or scenario grouping. This is another interesting direction for future research.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (No. 2021R1A2C2005531). The authors would like to thank the Institute for Industrial Systems Innovation of Seoul National University for the administrative support. We also are grateful to the editor and three anonymous referees, whose comments and suggestions enabled us to improve the paper significantly.

References

References

Alem, D., Curcio, E., Amorim, P., & Almada-Lobo, B. (2018). A computational study of the general lot-sizing and scheduling model under demand uncertainty via robust and stochastic approaches. *Computers & Operations Research*, *90*, 125–141.

Alfieri, A., Brandimarte, P., & D'Orazio, S. (2002). LP-based heuristics for the capacitated lot-sizing problem: The interaction of model formulation and solution algorithm. *International Journal of Production Research*, *40*, 441–458.

Alipour, Z., Jolai, F., Monabbati, E., & Zaerpour, N. (2020). General lot-sizing and scheduling for perishable food products. *RAIRO - Operations Research*, *54*, 913–931.

Almada-Lobo, B., Clark, A., Guimarães, L., Figueira, G., & Amorim, P. (2015). Industrial insights into lot sizing and scheduling modeling. *Pesquisa Operacional*, *35*, 439–464.

Almada-Lobo, B., Klabjan, D., Antónia Carravilla, M., & Oliveira, J. F. (2007). Single machine multi-product capacitated lot sizing with sequence-dependent setups. *International Journal of Production Research*, *45*, 4873–4894.

Almeder, C., & Almada-Lobo, B. (2011). Synchronisation of scarce resources for a parallel machine lotsizing problem. *International Journal of Production Research*, *49*, 7315–7335.

Belo-Filho, M. A. F., Toledo, F. M. B., & Almada-Lobo, B. (2014). Models for capacitated lot-sizing problem with backlogging, setup carryover and crossover. *Journal of the Operational Research Society*, *65*, 1735–1747.

Bertsimas, D., & Demir, R. (2002). An approximate dynamic programming approach to multidimensional knapsack problems. *Management Science*, *48*, 550–565.

Büyüktahtakın, İ. E., & Liu, N. (2016). Dynamic programming approximation algorithms for the capacitated lot-sizing problem. *Journal of Global Optimization*, *65*, 231–259.

Camargo, V. C. B., Toledo, F. M. B., & Almada-Lobo, B. (2012). Three time-based scale formulations for the two-stage lot sizing and scheduling in process industries. *Journal of the Operational Research Society*, *63*, 1613–1630.

Carvalho, D. M., & Nascimento, M. C. (2021). Hybrid matheuristics to solve the integrated lot sizing and scheduling problem on parallel machines with sequence-dependent and non-triangular setup. *European Journal of Operational Research*, .

Clark, A., Mahdieh, M., & Rangel, S. (2014). Production lot sizing and scheduling with non-triangular sequence-dependent setup times. *International Journal of Production Research*, *52*, 2490–2503.

Copil, K., Wörbelauer, M., Meyr, H., & Tempelmeier, H. (2017). Simultaneous lotsizing and scheduling problems: a classification and review of models. *OR Spectrum*, *39*, 1–64.

Federgruen, A., & Tzur, M. (1991). A simple forward algorithm to solve general dynamic lot sizing models with n periods in $o(n \log n)$ or $o(n)$ time. *Management Science*, *37*, 909–925.

Fiorotto, D. J., Huaccha Neyra, J. d. C., & de Araujo, S. A. (2020). Impact analysis of setup carryover and crossover on lot sizing problems. *International Journal of Production Research*, *58*, 6350–6369.

Fiorotto, D. J., Jans, R., & de Araujo, S. A. (2017). An analysis of formulations for the capacitated lot sizing problem with setup crossover. *Computers & Industrial Engineering*, *106*, 338–350.

Fleischmann, B., & Meyr, H. (1997). The general lotsizing and scheduling problem. *OR Spectrum*, *19*, 11–21.

Guimarães, L., Klabjan, D., & Almada-Lobo, B. (2014). Modeling lotsizing and scheduling problems with sequence dependent setups. *European Journal of Operational Research*, *239*, 644–662.

James, R. J., & Almada-Lobo, B. (2011). Single and parallel machine capacitated lotsizing and scheduling: New iterative mip-based neighborhood search heuristics. *Computers & Operations Research*, *38*, 1816–1825.

Koçlar, A., & Süral, H. (2005). A note on "The general lot sizing and scheduling problem". *OR Spectrum*, *27*, 145–146.

Krarup, J., & Bilde, O. (1977). Plant location, set covering and economic lot size: An $O(mn)$-algorithm for structured problems. In *Numerische Methoden bei Optimierungsaufgaben Band 3*.

Lee, Y., & Lee, K. (2020). Lot-sizing and scheduling in flat-panel display manufacturing process. *Omega*, *93*, 102036.

Maes, J., McClain, J. O., & Van Wassenhove, L. N. (1991). Multilevel capacitated lotsizing complexity and LP-based heuristics. *European Journal of Operational Research*, *53*, 131–148.

Mahdieh, M., Clark, A., & Bijari, M. (2018). A novel flexible model for lot sizing and scheduling with non-triangular, period overlapping and carryover setups in different machine configurations. *Flexible Services and Manufacturing Journal*, *30*, 884–923.

Melega, G. M., de Araujo, S. A., & Morabito, R. (2020). Mathematical model and solution approaches for integrated lot-sizing, scheduling and cutting stock problems. *Annals of Operations Research*, *295*, 695–736.

Menezes, A. A., Clark, A., & Almada-Lobo, B. (2011). Capacitated lot-sizing and scheduling with sequence-dependent, period-overlapping and non-triangular setups. *Journal of Scheduling*, *14*, 209–219.

Meyr, H. (2000). Simultaneous lotsizing and scheduling by combining local search with dual reoptimization. *European Journal of Operational Research*, *120*, 311–326.

Meyr, H. (2002). Simultaneous lotsizing and scheduling on parallel machines. *European Journal of Operational Research*, *139*, 277–292.

Meyr, H., & Mann, M. (2013). A decomposition approach for the general lotsizing and scheduling problem for parallel production lines. *European Journal of Operational Research*, *229*, 718–731.

Mohan, S., Gopalakrishnan, M., Marathe, R., & Rajan, A. (2012). A note on modelling the capacitated lot-sizing problem with set-up carryover and set-up splitting. *International Journal of Production Research*, *50*, 5538–5543.

Pochet, Y., & Wolsey, L. A. (2006). *Production planning by mixed integer programming*. Springer Science & Business Media.

Powell, W. B. (2007). *Approximate dynamic programming: Solving the curses of dimensionality*. John Wiley & Sons.

Powell, W. B. (2016). Perspectives of approximate dynamic programming. *Annals of Operations Research*, *241*, 319–356.

Suerie, C. (2006). Modeling of period overlapping setup times. *European Journal of Operational Research*, *174*, 874–886.

Suerie, C., & Stadtler, H. (2003). The capacitated lot-sizing problem with linked lot sizes. *Management Science*, *49*, 1039–1054.

Sung, C., & Maravelias, C. T. (2008). A mixed-integer programming formulation for the general capacitated lot-sizing problem. *Computers & Chemical Engineering*, *32*, 244–259.

Wagner, H. M., & Whitin, T. M. (1958). Dynamic version of the economic lot size model. *Management Science*, *5*, 89–96.

Wolsey, L. A. (1997). MIP modelling of changeovers in production planning and scheduling problems. *European Journal of Operational Research*, *99*, 154–165.

Appendix A. Proof of Proposition 3.2

For a given $X \in \mathcal{X}$, by summing the constraints (2c) over all $(i,s) \in X$ and eliminating the common terms of both sides, we obtain

$$\sum_{(i,s)\in X} \left(a_i x_{is} + r_{is} + \sum_{j\in\mathcal{I}} st_{ji} z_{jis} \right) = \sum_{(i,k,s+1)\in\delta^+(X)} w_{iks+1} - \sum_{(j,i,s)\in\delta^-(X)} w_{jis}.$$

From the bound constraints (2h) and bound values (3), we have

$$\sum_{(i,s)\in X}\Big(a_i x_{is}+\sum_{j\in\mathcal{I}} st_{ji}z_{jis}\Big) \le \sum_{(i,k,s+1)\in\delta^+(X)} U_{iks+1}z_{iks+1} - \sum_{(j,i,s)\in\delta^-(X)} L_{jis}z_{jis}$$
$$= \sum_{\substack{(i,k,s+1)\in\delta^+(X):\\ s+1=f^{T(s+1)}}} start_{T(s+1)}z_{iks+1} + \sum_{\substack{(i,k,s+1)\in\delta^+(X):\\ s+1\neq f^{T(s+1)}}} (end_{T(s+1)}-st_{ik})z_{iks+1}$$
$$- \sum_{\substack{(j,i,s)\in\delta^-(X):\\ s=f^{T(s)}}} (start_{T(s)}-st_{ji})z_{jis} - \sum_{\substack{(j,i,s)\in\delta^-(X):\\ s\neq f^{T(s)}}} start_{T(s)}z_{jis}.$$

As $start_{T(s+1)} = end_{T(s)}$ when $s+1 = f^{T(s+1)}$ and $end_{T(s+1)} = end_{T(s)}$ when $s+1 \neq f^{T(s+1)}$, the right-hand side term is equal to

$$\sum_{(i,k,s+1)\in\delta^+(X)} end_{T(s)}z_{iks+1} - \sum_{(j,i,s)\in\delta^-(X)} start_{T(s)}z_{jis} - \sum_{\substack{(i,k,s+1)\in\delta^+(X):\\ s+1\neq f^{T(s+1)}}} st_{ik}z_{iks+1} + \sum_{\substack{(j,i,s)\in\delta^-(X):\\ s=f^{T(s)}}} st_{ji}z_{jis}.$$

After the rearrangement of the terms, the inequalities (4) can be derived. In addition, when X is restricted to a subset of $\mathcal{X}_t$, $T(s) = t$. Moreover, $\sum_{(i,k,s+1)\in\delta^+(X)} z_{iks+1} = \sum_{(j,i,s)\in\delta^-(X)} z_{jis}$. Therefore, inequalities (5) can be obtained.

Appendix B. Facility Location Reformulation

The following additional variables and notations are introduced for the reformulations `(ST-FL)` and `(TF-FL)`. We define a dummy microperiod $S+1$ to represent the demand that is not satisfied. Let $T(S+1) = T+1$ and $\mathcal{S}^0 = \mathcal{S}\cup\{S+1\}$. Then, we define variable α_{ist} as the fraction of demand of item i in macroperiod t satisfied by the production in microperiod s. If $s = S+1$, it represents the fraction of demand of item i in macroperiod t that is not satisfied.

Regarding the cost, let $H_{ist} = \sum_{k=T(s)}^{t-1} hc_{ik}d_{it}$ when $T(s) < t$; otherwise, $H_{ist} = 0$. Similarly, let $B_{ist} = \sum_{k=t}^{T(s)-1} bc_{ik}d_{it}$ when $T(s) > t$; otherwise, $B_{ist} = 0$. The cost coefficient of α_{ist} is defined as $C_{ist} := H_{ist} + B_{ist} + pc_{iT(s)}d_{it}$. The `(ST-FL)` is as follows:

$$\text{minimize} \quad \sum_{i\in\mathcal{I}}\sum_{t\in\mathcal{T}}\Big(\sum_{s\in\mathcal{S}^0} C_{ist}\alpha_{ist} + \sum_{j\in\mathcal{I}}\sum_{s\in\mathcal{S}_t} sc_{ijt}z_{ijs}\Big) \tag{B.1a}$$

$$\text{subject to} \quad \sum_{s\in\mathcal{S}^0}\alpha_{ist} = 1 \qquad \forall i\in\mathcal{I}, t\in\mathcal{T} \tag{B.1b}$$

$$\sum_{i\in\mathcal{I}}\sum_{s\in\mathcal{S}_t}\Big(\sum_{l\in\mathcal{T}} a_i d_{il}\alpha_{isl} + \sum_{j\in\mathcal{I}} st_{ij}z_{ijs}\Big) \le K_t + \sum_{i\in\mathcal{I}}\sum_{j\in\mathcal{I}}(v_{ijt} - v_{ijt+1}) \qquad \forall t\in\mathcal{T} \tag{B.1c}$$

$$\alpha_{ist} \le y_{is} \qquad \forall i\in\mathcal{I}, s\in\mathcal{S}, t\in\mathcal{T} \tag{B.1d}$$

$$\sum_{t\in\mathcal{T}} a_i d_{it}\alpha_{ist} \le K_{T(s)}y_{is} \qquad \forall i\in\mathcal{I}, s\in\mathcal{S} \tag{B.1e}$$

$$\sum_{i\in\mathcal{I}} y_{i1} = 1 \tag{B.1f}$$

$$y_{is} = \sum_{j\in\mathcal{I}} z_{jis} \qquad \forall i\in\mathcal{I}, s\in\mathcal{S}\setminus\{1\} \tag{B.1g}$$

$$y_{is} = \sum_{j \in \mathcal{I}} z_{ijs+1} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S} \setminus \{S\} \quad \text{(B.1h)}$$

$$v_{ijt} \leq st_{ij} q_{ijt} \qquad \forall i, j \in \mathcal{I}, t \in \mathcal{T} \quad \text{(B.1i)}$$

$$q_{ijt} \leq z_{ijf^t} \qquad \forall i, j \in \mathcal{I}, t \in \mathcal{T} \quad \text{(B.1j)}$$

$$\alpha_{ist},\ y_{is},\ v_{ijt} \geq 0 \qquad \forall i, j \in \mathcal{I}, t \in \mathcal{T}, s \in \mathcal{S}^0 \quad \text{(B.1k)}$$

$$q_{ijt}, z_{ijs} \in \{0, 1\} \qquad \forall i, j \in \mathcal{I}, t \in \mathcal{T}, s \in \mathcal{S} \quad \text{(B.1l)}$$

The objective function (B.1a) is the total cost. Constraints (B.1b) are the demand constraints. Constraints (B.1c) are the capacity constraints. Constraints (B.1d) indicate that an item can be produced only if the corresponding setup occurs. Constraints (B.1e) represent the upper bound constraints for the amount of an item produced in a microperiod. Constraints (B.1f)–(B.1l) are defined in the same manner as (1e)–(1k) of `(ST)`, except for the domains of variable $\boldsymbol{\alpha}$. Similarly, the `(TF-FL)` is as follows:

$$\text{minimize} \quad \sum_{i \in \mathcal{I}} \sum_{t \in \mathcal{T}} \Big(\sum_{s \in \mathcal{S}^0} C_{ist} \alpha_{ist} + \sum_{j \in \mathcal{I}} \sum_{s \in \mathcal{S}_t} sc_{ijt} z_{ijs} \Big) \quad \text{(B.2a)}$$

$$\text{subject to} \quad \sum_{s \in \mathcal{S}^0} \alpha_{ist} = 1 \qquad \forall i \in \mathcal{I}, t \in \mathcal{T} \quad \text{(B.2b)}$$

$$\sum_{j \in \mathcal{I}} (w_{jis} + st_{ji} z_{jis}) + \sum_{l \in \mathcal{T}} a_i d_{il} \alpha_{isl} + r_{is} = \sum_{k \in \mathcal{I}} w_{iks+1} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S} \quad \text{(B.2c)}$$

$$\alpha_{ist} \leq y_{is} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S}, t \in \mathcal{T} \quad \text{(B.2d)}$$

$$r_{is} + \sum_{t \in \mathcal{T}} a_i d_{it} \alpha_{ist} \leq K_{T(s)} y_{is} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S} \quad \text{(B.2e)}$$

$$\sum_{i \in \mathcal{I}} y_{i1} = 1 \quad \text{(B.2f)}$$

$$y_{is} = \sum_{j \in \mathcal{I}} z_{jis} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S} \setminus \{1\} \quad \text{(B.2g)}$$

$$y_{is} = \sum_{j \in \mathcal{I}} z_{ijs+1} \qquad \forall i \in \mathcal{I}, s \in \mathcal{S} \setminus \{S\} \quad \text{(B.2h)}$$

$$L_{ijs} z_{ijs} \leq w_{ijs} \leq U_{ijs} z_{ijs} \qquad \forall i, j \in \mathcal{I}, s \in \mathcal{S} \setminus \{1\} \quad \text{(B.2i)}$$

$$\alpha_{ist},\ y_{is},\ w_{ijs} \geq 0 \qquad \forall i \in \mathcal{I}, s \in \mathcal{S}^0, t \in \mathcal{T} \quad \text{(B.2j)}$$

$$z_{ijs} \in \{0, 1\} \qquad \forall i, j \in \mathcal{I}, s \in \mathcal{S} \quad \text{(B.2k)}$$

The objective function (B.2a) and other constraints (B.2b) and (B.2d)–(B.2k) are defined in the same manner as those for `(ST-FL)`. Constraints (B.2c) are the same as the time flow balance equations of `(TF)`, except that variable x_{is} is replaced with $\sum_{l \in \mathcal{T}} d_{il} \alpha_{isl}$.