processes

MDPI

Article

Capacitated Lot-Sizing Problem with Sequence-Dependent Setup, Setup Carryover and Setup Crossover

Jangha Kang

Department of Industrial Engineering, Chosun University, 309 Pilmun-Daero, Dong-Gu, Gwangju 61452, Korea; janghakang@chosun.ac.kr; Tel.: +82-10-3901-4601

Received: 30 May 2020; Accepted: 3 July 2020; Published: 5 July 2020

Abstract: Since setup operations have significant impacts on production environments, the capacitated lot-sizing problem considering arbitrary length of setup times helps to develop flexible and efficient production plans. This study discusses a capacitated lot-sizing problem with sequence-dependent setup, setup carryover and setup crossover. A new mixed integer programming formulation is proposed. The formulation is based on three building blocks: the facility location extended formulation; the setup variables with indices for the starting and the completion time periods; and exponential number of generalized subtour elimination constraints (GSECs). A separation routine is adopted to generate the violated GSECs. Computational experiments show that the proposed formulation outperforms models from the literature.

Keywords: capacitated lot-sizing; sequence-dependent setup; setup carryover; setup crossover; branch-and-cut algorithm

1. Introduction

1.1. Background

Master production plans of a main production process of a supply chain provide all the participants with the visibility regarding the operation. Production planning is referred to as "lot-sizing" in a discrete production environment [1]. The capacitated lot-sizing problem (CLSP) has been considered in various industries to generate production plans reflecting production restriction [1,2].

Generally, lot-sizing problem assumes that planning horizon is finite and can be divided into multiple time periods, which are terms as "time buckets". The time buckets are classified into small- and big-time buckets, when the comparison of the relative length of the time period is made with respect to the relative length of the production lot [3]. Models considering small-time buckets allow at most one or two products to be produced in a period and models considering big-time buckets allow more than two products to be produced in a period [1].

Setup operations prepare processing units (e.g., materials and machines) to manufacture production lots. In process industries, the setup time required for replacing production models accounts for a large proportion of capacity utilization. Moreover, variations in the setup time may be highly depending on the product models and their operating sequence. The setup time between similar products is relatively short, allowing engineers to setup multiple times in a day or a shift. However, setup operations requiring more than one day or several days may be necessary between products when the processing material or equipment needs to be changed. For solving the big-time bucket CLSP model, several modeling techniques have been developed to reflect these characteristics of setup time. Sequence dependent setup time model is used to reflect setup time that depends on product order.

Several studies [4,5] have introduced CLSP models, in which the setup state can be preserved across several periods and have used various terms such as "linked lot sizes" and "setup carryover". In this paper, the concept is called as setup carryover. The setup carryover allows a setup state to be maintained from one period to the following one—namely, a setup carryover model enables a product to be produced in the following buckets without additional setup time. In this concept, production planers in process industries often prefer to produce only a single product without model change during several days or shifts.

In some manufacturing environments, the setup started in period t can be split between the periods t and $t+1$ [6,7]. In this paper, the concept is called as "setup-splitting".

In most continuous manufacturing industries such as process industries, due to the presence of long setup times, manufacturers allow the setup to be carried across more than one period [1], the concept is called as "setup crossover". Setup crossover enables the incomplete setup operation to cross over the boundaries of time periods. In the case that setup times are longer than a period length, the setup operation may be performed in more than two periods. However, if all the setup times are less than a period, the setup crossover is exactly same as the setup-splitting.

1.2. Literature Review

There are several studies published on big-bucket CLSP with setup times. However, studies on CLSPs with setup carryover are limited. Sox and Gao [4] presented two formulations for the CLSP with setup carryover. Suerie and Stadtler [5] presented a formulation for the CLSP with setup carryover and their computational tests have shown that their formulation is better than the formulation provided by [4]. Further, Menezes et al. [6] proposed CLSPs with setup-splitting considering sequence-dependent and non-triangular setups. In the formulation, the classical subtour elimination constraints, known as the Miller–Tucker–Zemlin (MTZ) constraints, introduced by Dantzig et al. [8] are used. Kopanos et al. [9] introduced a CLSPs with setup carryover and setup-splitting problem with items classified into product families. They considered sequence-dependent setups between products in different families and sequence-independent setups between products in the same family. Mohan et al. [10] extended the formulation of Suerie [5] to address setup-splitting. Camargo et al. [11] presented three formulations for the two-stage lot-sizing and scheduling problem. With computational tests, they showed that the formulation with setup-splitting is the most flexible to incorporate setup-related features. Ramya et al. [7] proposed a formulation for the CLSP with setup carryover and setup-splitting and no backorders or lost sales. Fiorotto et al. [12] compared two formulations proposed by Mohan [10] and by Menezes [6] and proposed symmetry braking constraints. With computational experiments, they showed that the proposed constraints are effective for the formulation proposed by Mohan.

Sung and Maravelias [13], Belo-Filho et al. [14] considered the CLSPs with setup carryover and setup crossover with the setup operations can be split into more than two periods. Sung and Maravelias [13] proposed a big-bucket formulation. Belo-Filho et al. [14] proposed two formulations for the case with backlog. One of the formulations involves a time index disaggregation, defining the setup variables with indices for the starting and the completion time periods of the setup operation.

For the case of sequence dependent setup, two groups of studies have been published. First group of studies consider the problems where at most one lot could be generated per a product for each time bucket. The other group of studies allow multiple lots per a product for each time bucket. The former problems usually consider triangular setup times and costs, while the latter problems usually consider non-triangular setup times or minimum lot sizes. Almada-Lobo et al. [15] presented two models for the CLSP with sequence-dependent and triangular setup times and costs using the MTZ subtour prohibition constraints. Clark et al. [16] formulated a sequencing and lot-sizing model with non-triangular setup times based on the asymmetric traveling salesman problem. Ramya et al. [1] presented two models based on MTZ subtour prohibition constraints for the CLSP with sequence-dependent setup times and setup cost and with setup carryover and setup crossover. In the model, no idle times are allowed

during setup operation and production of a product. In the models of [1,15,16] at most one lot per product can be produced in a time period.

Menezes et al. [6] allowed production of multiple lots per period. Clark et al. [17] presented a stronger formulation than Menezes et al. [6] for modeling the production of multiple many a product per period by using a polynomial number of multicommodity-flow-type constraints [18]. Mahdieh et al. [19] presented formulations based on Clark's formulation for lot-sizing and scheduling with minimum lot-size, non-triangular sequence-dependent setup times and costs with setup carryover and setup-splitting.

1.3. Contribution and Organization

In this paper, a CLSP with sequence-dependent and triangular setup times and with setup carryover and setup crossover is considered. For the problem, a new mathematical formulation is proposed. The formulation is based on the facility location extended formulation [2]. Moreover, it uses the setup variables with indices for the starting and the completion time periods as Belo-Filho's formulation [14] and exponential number of generalized subtour elimination constraints (GSECs) to prevent subtours in a time bucket.

The mixed integer programming (MIP) formulations are usually compared in terms tightness or compactness [20]. The tightness of an MIP formulation is defined as the difference between the optimal values for the continuous relaxation and the original MIP problem. Tightening an MIP formulation, usually by adding cutting planes or valid inequalities, can reduce the search space that the solver needs to explore [21]. Meanwhile, the compactness refers to the quantity of data that must be processed when solving the problem [20]. Because there are many relaxations that must be solved when solving an MIP problem, the compactness of such a problem always refers to the size of these relaxations. A more compact formulation can speed up the search for the optimal solutions. The tightness and the compactness of the formulation are shown by comparison with one of Mahdieh's formulation (deferred to as MLOV-SM) [19] and one of Ramya's formulation (deferred to as MM2) [1].

The remainder of the paper is organized as follows. In Section 2, a formal description of the problem is given. In Section 3, the formulation for the problem is presented. The formulation has exponential numbers of the GSECs. The separation algorithms for the GSECs are explained in Section 4. In addition to comparison with other models, computational experiments and their results are reported in Section 5. Finally, Section 6 concludes this study and suggests some directions for further research.

During the computational experiments, the author found that the setup-splitting is not implemented properly in MLOV-SM. The modified and fixed version of MLOV-SM is reported in Appendix A.

2. Problem Statement

In the following, a formal description of the problem is presented with assumptions for clarifying and simplifying the mathematical model.

A planning horizon consists of finite number of time buckets. For each bucket, an index ranging from 1 to n is assigned; that is, the number of buckets is denoted by n. For each bucket $t = 1, \cdots, n$, a positive amount of capacity denoted by C_t is assigned. In the problem, the production lot sizes and schedules for a set of items, denoted by I, are planned. For each item i in I and bucket t, a non-negative demand quantity denoted by d_t^i is given. The demand could be satisfied through production before the due date with inventory holding cost h_t^i for each bucket or after the due date with backlogging cost b_t^i. Production of one unit of items i in bucket t consumes a positive amount of capacity denoted by p_t^i, namely the process time per unit. All the demands must be met until the end of the planning horizon. Let T_i denote the set of buckets where a positive amount of demands is defined for item i. Considering sequence-dependent setup time and cost, production sequences were decided for each bucket. Production change from item i to j requires setup time τ_{ij} and causes cost c_{ij}. If capacity allows, multiple setup could occur in a bucket. With long setup time or inadequate capacities, setup operation

could be performed during consecutive multiple buckets. Production of the last lot could resume in the following bucket without any burden of setup operations. The initial and final setup states are not given. The objective is to minimize the overall cost, which includes backlogging, inventory holding and setup costs.

Under the above descriptions, we can introduce the following two assumptions without loss of generality.

Assumption 1. *No idle time is allowed during setup operations.*

Bucket-independent setup times and costs were considered, therefore, any idle during setup operations could be moved to the buckets before setup started or after setup finished without increasing the cost. Therefore, the Assumption 1 is acceptable.

Assumption 2. *In a bucket, at most one lot can be generated for an item.*

In [14], the setup variables with indices for the starting and completion time periods of the setup operation were introduced to solve CLSP with setup carryover and setup crossover. According to Assumption 1, for each type of setup (for each item in [14]) with setup time τ, a setup variable can be defined for each ordered bucket pair (s, t) satisfying the following constraint:

$$1 \le s \le t \le n \text{ and } \sum_{u=s+1}^{t-1} C_u < \tau \le \sum_{u=s}^{t} C_u \tag{1}$$

Proposition 1. *The number of ordered bucket pairs satisfying constraints (1) are limited by* $2n - 1$. (Proof: refer to [14]).

For a sequence-dependent setup operation from item i to j, we can consider the setup variable indexed by bucket pairs in the following set:

$$T^{ij} \equiv \left\{ (s,t) \middle| 1 \le s \le t \le n, \sum\nolimits_{u=s+1}^{t-1} C_u < \tau_{ij} \le \sum\nolimits_{u=s}^{t} C_u \right\} \tag{2}$$

3. Formulation

The problem formulation is based on three building blocks: the facility location extended formulation [2]; the setup variables with indices for the starting and the completion time buckets [14]; and exponential number of generalized subtour elimination constraints (GSECs) [18].

The following two variables come from the facility location extended formulation: setup state variable, y_t^i, for item i and bucket t; and production portion variable, x_{tu}^i, for item i and bucket t and bucket $u \in T_i$ (having positive demand d_u^i). y_t^i has value one if setup states for item i in bucket t is on and item i can be produced and 0 otherwise. Moreover, x_{tu}^i denotes the proportion of the demand d_u^i produced in bucket t. For example, in the case $\left(d_2^i, d_3^i\right) = (50, 100)$, $\left(y_1^i, y_2^i, y_3^i, x_{12}^i, x_{13}^i, x_{23}^i\right) = (1, 1, 0, 1, 0.5, 0.5)$ means that total demand of d_2^i and half of demand d_3^i are produced in bucket 1 and the remaining half of demand d_3^i is produced in bucket 2.

In the formulation, model change or setup operation is indicated by the following setup-indicating variable: e_{st}^{ij}, for a pair of items $i \ne j \in I$ and an ordered pair of buckets $(s, t) \in T^{ij}$. The variable has a value of one if setup operation from item i to j starts in bucket s and finishes in bucket t and the value is zero otherwise. If $s < t$, e_{st}^{ij} indicates setup crossover. When $s + 1 < t$, e_{st}^{ij} indicates setup crossover longer than two buckets. By Proposition 1, the number of the variables (e_{st}^{ij}) is bounded by $(2n - 1) \times |I| \times (|I| - 1)$. R_t^i denotes the remaining setup times for item i at the beginning of bucket t in the case of setup crossover. In the formulation, a setup carryover indicating variable, z_t^i, is used for each item i and bucket $t = 1, \cdots, n + 1$. z_t^i is has value of one if the setup state for item i is maintained at the

boundary between bucket $t-1$ and t and zero otherwise. As mentioned in the problem statement, the initial and finial setup states are not given. Hence, z_1^i's decide initial setup state and z_{n+1}^i's decide final setup state.

Lot sequencing with GSECs needs exponential number of constraints, but do not need any additional variables. The process of deriving the constrains is introduced at the end of this section. Table 1 presents the parameters and the variables used in the formulation.

Table 1. Notation.

Types	Notation	Meaning
Set		
	I	Set of items, index $i,\ j,\ k$
	T	Set of buckets, $\{1, 2, \cdots, n\}$
	T_i	Set of buckets where a positive amount of demands is defined for item i
	T^{ij}	Set of ordered pair of buckets where the first means the starting bucket of the setup operation from i to j and the second means the ending bucket of the setup
Parameters		
	n	Number of buckets, index s, t, u
	C_t	Capacity of bucket $t = 1, \cdots, n$
	d_t^i	Demand of item i in bucket t
	p_t^i	Process time per unit of item i in bucket t
	h_t^i	Inventory holding cost of item i in bucket t
	b_t^i	Backlogging cost of item i in bucket t
	τ_{ij}	Setup time from item i to j
	c_{ij}	Setup cost from item i to j
Variables		
	x_{tu}^i	Proportion of the demand for item i in bucket u produced in bucket t, for $i \in I, u \in T^i$
	y_t^i	Variable indicating setup state: 1 if setup states for item i in bucket t is on and item i can be produced, and 0 otherwise.
	z_t^i	Variable indicating setup carryover: 1 if the setup state is for item i at the beginning of bucket t, and 0 otherwise
	e_{st}^{ij}	Variable indicating setup change: 1 if setup operation from item i to item j starts in bucket s and finishes in bucket t, and 0 otherwise, for $i \neq j \in I, \in$
	R_t^i	Remaining setup times for item i at the beginning of bucket t in the case of setup crossover
	L_t	Idle time in bucket t

The formulation M1 is as follows:

$$\textbf{Min} \sum_{i \in I} \sum_{t \in T_i} \left(\sum_{s=1}^{t-1} \sum_{u=s}^{t-1} h_u^i d_t^i x_{st}^i + \sum_{s=t+1}^{n} \sum_{u=t}^{s-1} b_u^i d_t^i x_{st}^i \right) + \sum_{i,j \in I | i \neq j} \sum_{(s,t) \in T^{ij}} c_{ij} e_{st}^{ij} \tag{3}$$

Subject to

$$\sum_{s=1}^{n} x_{st}^i = 1, \text{ for all } i \in I, t \in T_i \tag{4}$$

$$x_{st}^i \leq y_s^i, \text{ for all } i \in I, t \in T_i, s = 1, \cdots, n \tag{5}$$

$$\sum_{u \in T_i} p_t^i d_u^i x_{tu}^i \leq C_t \cdot y_t^i, \text{ for all } i \in I, t = 1, \cdots, n \tag{6}$$

$$\sum_{i,j \in I | i \neq j} \sum_{(t,u) \in T^{ij}} \tau_{ij} e_{tu}^{ij} + \sum_{i \in I} \sum_{u \in T_i} p_t^i d_u^i x_{tu}^i + \sum_{i \in I} R_t^i + L_t = C_t + \sum_{i \in I} R_{t+1}^i, \text{ for all } t = 1, \cdots, n \tag{7}$$

$$\sum_{i\in I} z_t^i + \sum_{i,j\in I|i\neq j} \sum_{(s,u)\in T^{ij}|s<t,u\geq t} e_{su}^{ij} \leq 1, \text{ for all } t = 1,\cdots,n+1 \tag{8}$$

$$z_t^j + \sum_{i\in I|i\neq j} \sum_{(s,t)\in T^{ij}} e_{st}^{ij} = y_t^j, \text{ for all } j \in I, t = 1,\cdots,n \tag{9}$$

$$z_{t+1}^i + \sum_{j\in I|i\neq j} \sum_{(t,u)\in T^{ij}} e_{tu}^{ij} = y_t^i, \text{ for all } i \in I, t = 1,\cdots,n \tag{10}$$

$$R_t^j \leq \sum_{i\in I|i\neq j} \sum_{(s,u)\in T^{ij}|s<t,t\leq u} \tau^{ij} e_{su}^{ij}, \text{ for all } j \in I, t = 2,\cdots,n \tag{11}$$

$$\begin{aligned} &\sum_{i,j\in I|i\neq j} \sum_{(t,t)\in T^{ij}} \tau^{ij} e_{tt}^{ij} + \sum_{i\in I} \sum_{u\in T_i} p_t^i d_u^i x_{tu}^i + L_t \\ &\leq C_t \cdot \left(1 - \sum_{i,j\in I|i\neq j} \sum_{(s,u)\in T^{ij}|s<t,t<u} e_{su}^{ij}\right), \text{ for all } t = 2,\cdots,n-1 \end{aligned} \tag{12}$$

$$\sum_{i\neq j\in S|(t,t)\in T^{ij}} e_{tt}^{ij} \leq \sum_{i\in S\setminus\{k\}} y_t^i, \text{ for all } S \subset I, k \in S, t = 1,\cdots,n \tag{13}$$

$$e_{st}^{ij} \in \{0,1\} \text{ for all } i \neq j \in I, \ (s,t) \in T^{ij} \tag{14}$$

$$z_t^i \in \{0,1\} \text{ for all } i \in I, t = 1,\cdots,n+1 \tag{15}$$

$$x_{st}^i \geq 0 \text{ for all } i \in I, t \in T_i, s \in T \tag{16}$$

$$y_t^i \in \{0,1\} \text{ for all } i \in I, t \in T \tag{17}$$

$$L_t \geq 0 \text{ for all } t \in T \tag{18}$$

$$R_t^i \geq 0 \text{ for all } i \in I, t = 2,\cdots,n, \ R_1^i = R_{n+1}^i = 0 \text{ for all } i \in I. \tag{19}$$

Here, the objective function (3) minimizes the total cost including backlogging, inventory holding and setup cost. Constraints (4) ensure that each demand is met within the planning horizon. Constraints (5) and (6) ensure that production occurs only in the bucket where a lot is generated. Constraints (7) ensure that capacity consumption during production and setup change plus the idle time equal to the capacity of each bucket, considering the remaining setup times during setup crossover. Constraints (8) ensure that setup state is dedicated to at most one item at the beginning of a bucket. Constraints (9) ensure that a lot comprising one item is generated in a bucket when the setup of the item is carried over or a setup operation to the item ends in the bucket. Constraints (10) ensure that one lot is followed by a setup carryover operation to the next bucket or a setup change operation to another item. Constraints (11) ensure that remaining setup times are allowed only during setup crossover. From Assumption 1, idle time is not allowed during setup operation, this is reflected in constraints (12). First two terms of the left-hand side are added to lift the constraint. Constraints (13) prevent the subtours in each bucket. Constraints (14)–(19) describe the domains of the variables. From Assumption 2, the number of lots is limited to one for each item and bucket and this is reflected in constraints (17). In the formulation, $(2n-1)\times|I|^2 + 2\times|I|$ binary integer variables, and $n\times(\sum_{i\in I}|T_i| + |I| + 1)$ fractional variables are defined. Moreover, total number of constraints except constraints (13) is $(n+1)\times\sum_{i\in I}|T_i| + 4n\times|I| + (n-1)\times|I| + 3n - 1$.

In the remainder of this section, the process of deriving the constrains (13) is introduced. Constraints (9), (10) and the variables in the constraints can be converted into a network as Figure 1; here, the constraints and variables are converted to nodes and arcs, respectively. Without constraints (13), a sequence-dependent setup usually causes subtours, indicated by the circle with arcs y_t^i, e_{tt}^{ij}, y_t^j, and e_{tt}^{ji} in the figure. To discard such a solution, subtour elimination constraints are used.

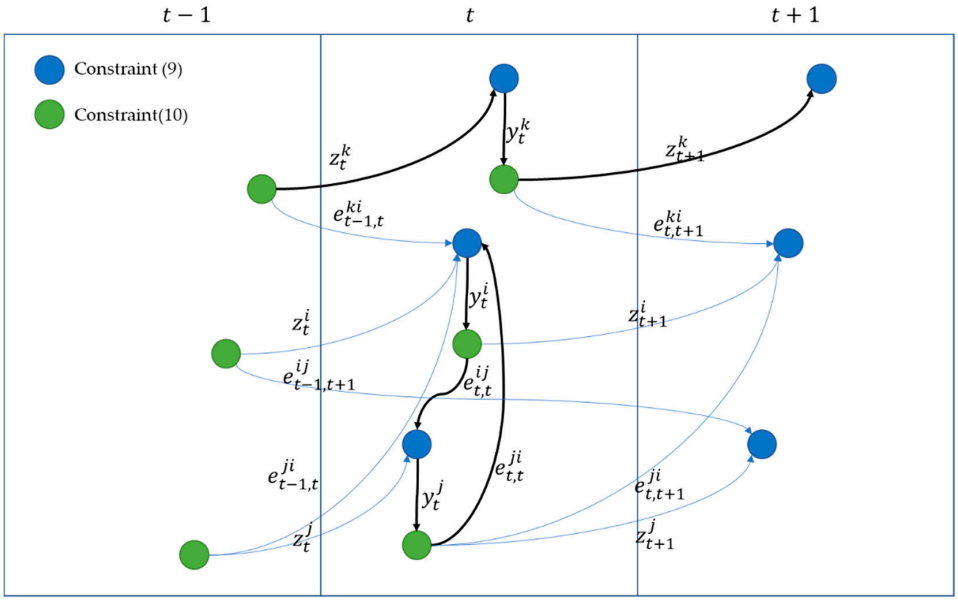

Figure 1. Illustration of a network with setup related constrains and variables and a subtour. (10) Green nodes denote constraints and (9) blue nodes denote constraints. Variables corresponding to thick arcs have values of one, and the others have values of zero. Circular setup change between (i, j) generates a subtour.

There are many studies on subtour elimination [18]. To obtain the required results, let us consider a converted network $\widetilde{G}_t(V_t, A_t)$ for each bucket t, as depicted in Figure 2. We generated the network using the following procedure:

1. contracting of the arcs relating to y's and merging the end nodes into one, denoted by y_t^i;
2. merging the tail nodes of arcs crossing the border between bucket $t-1$ and t into one node, denoted by "source";
3. and merging the head nodes of arcs crossing the border between bucket t and $t+1$ into one node, denoted by "sink".

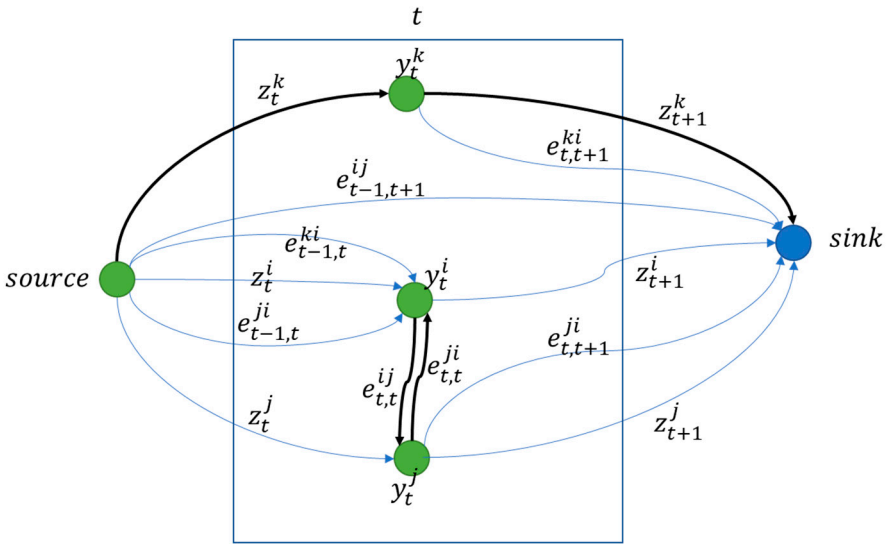

Figure 2. Illustration of the converted network $\widetilde{G}_t(V_t, A_t)$ for bucket t. Variables corresponding to thick arcs have values of one and the others have values of zero. Circular setup change between (i, j) generates a subtour.

Therefore, we obtained the followings:

$$V_t \equiv \left\{ y_t^i, i \in I \right\} \cup \{source,\ sink\} \tag{20}$$

$$\begin{aligned} A_t \equiv \left\{ z_t^i, i \in I \right\} \cup \left\{ z_{t+1}^i, i \in I \right\} \cup \left\{ e_{tt}^{ij}, i \neq j \in I, (t,t) \in T^{ij} \right\} \cup \left\{ e_{st}^{ij}, i \neq j \in I, (s,t) \in T^{ij} \middle| s < t \right\} \\ \cup \left\{ e_{tu}^{ij}, i \neq j \in I, (t,u) \in T^{ij} \middle| t < u \right\} \cup \left\{ e_{su}^{ij}, i \neq j \in I, (s,u) \in T^{ij} \middle| s < t < u \right\} \end{aligned} \tag{21}$$

Balas [22] introduced the prize-collecting traveling salesman problem to derive a model for scheduling the daily operation of a steel rolling mill. For the problem, Balas proposed the following generalized subtour elimination constraint:

$$\sum_{(i,j) \in \delta^+(S)} e^{ij} \geq \sum_{(k,j) \in \delta^+(k)} e^{kj} \text{ for all } k \in S, S \subset V \backslash \{source, sink\}, \tag{22}$$

where $e^{ij} = 1$ if setup changes from item i to j, 0 otherwise; $\delta^+(S)$ is the set of outgoing arcs from S. Constraints (22) ensure that the number of selected outgoing arcs from S is at least equal to the number of selected arcs going out from any node k in S. Consequently, subtours can be prevented.

Tacaari [18] rewrote constraints (22) as follows:

$$\sum_{(i,j) \in A(S)} e^{ij} \leq \sum_{i \in S \backslash \{k\}} \sum_{(i,j) \in \delta^+(i)} e^{ij} \text{ for all } k \in S, S \subset V \backslash \{source, sink\}, \tag{23}$$

where $A(S)$ is the set of arcs between two nodes in S. In $\widetilde{G}_t(V_t, A_t)$, S $\subset \left\{ y_t^i, i \in I \right\}$ and $A(S) \subset \left\{ e_{tt}^{ij}, i \neq j \in I \middle| (t,t) \in T^{ij} \right\}$; therefore, by replacing the $\sum_{(i,j) \in \delta^+(i)} e^{ij}$ with y_t^i (this relation comes from constraints (10)), we can rewrite constraint (23) to fit into our formulation as follows:

$$\sum_{i \neq j \in S | (t,t) \in T^{ij}} e_{tt}^{ij} \leq \sum_{i \in S \backslash \{k\}} y_t^i, \text{ for all } S \subset I, k \in S \tag{24}$$

Through generating constraints (24) for each bucket t, we can derive constraints (13), the subtour elimination constraint.

4. Branch-and-Cut Algorithm

There are exponential numbers of subsets $S \subset I$; therefore, there are exponential numbers of constraints (13) in the formulation. To solve a formulation with exponential number of constraints, we usually use the branch-and-cut algorithm. In the algorithm, most of the constraints are outside of the running formulation and constraints violated by the current solution of linear programming (LP) relaxation are identified dynamically and added into the running formulation. Therefore, we can control the size of the running formulation. The procedure of identifying the violated constraints is referred to as separation (details regarding separation and branch-and-cut algorithm are available in [21]).

4.1. Separation

After obtaining an optimal solution to the LP relaxation (let $\overline{e}_{st}^{ij}, \overline{y}_t^i$ denote the values of the variables e_{st}^{ij}, y_t^i, respectively, in the optimal solution) of the current running formulation, for each bucket $t = 1, \cdots, n$, we use two separation routines to find the violated subtour elimination constraints. One routine is based on the minimum cut, and the other is based on the strongly connected components.

In the routine based on the minimum cut, we generate a network $\overline{G}_t\left(\overline{V}_t, \overline{A}_t\right)$, illustrated in Figure 3, where $\overline{V}_t \equiv \left\{y_t^i, i \in I \middle| \overline{y}_t^i > 0\right\} \cup \{sink\}$ and $\overline{A}_t \equiv A_t^1 \cup A_t^2$. Here, A_t^1, A_t^2 are defined as follows:

$$A_t^1 \equiv \left\{e_{tt}^{ij}, i \neq j \in I \middle| (t,t) \in T^{ij}, \overline{e}_{tt}^{ij} > 0\right\}, \tag{25}$$

$$A_t^2 \equiv \left\{\left(y_t^i, sink\right) \middle| i \in I, \overline{y}_t^i > 0\right\}. \tag{26}$$

For each arc $e_{tt}^{ij} \in A_t^1$, we assign capacity $\overline{e}_{tt}^{ij}$ and for each arc $\left(y_t^i, sink\right) \in A_t^2$, we assign capacity $\overline{y}_t^i - \sum_{\forall j | e_{tt}^{ij} \in A_t^1} \overline{e}_{tt}^{ij}$.

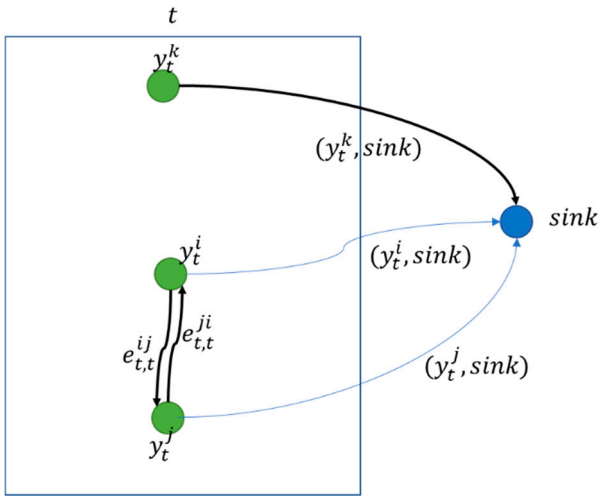

Figure 3. Illustration of a network $\overline{G}_t\left(\overline{V}_t, \overline{A}_t\right)$ for bucket t for the use of cut-based separation.

For each vertex $y_t^i \in \overline{V}_t \backslash \{sink\}$, the minimum cut between y_t^i and $sink$ yields a node partition $\left(S, \overline{V}_t \backslash S\right)$, where $y_t^i \in S$ and $sink \in \overline{V}_t \backslash S$. If the total capacity of the arcs from S to $\overline{V}_t \backslash S$ is less than $\overline{y}_t^i$, the constraint (24) with $k = y_t^i$ is violated by the current solution. Thus, we add the constraint into the running formulation. For each time bucket $t = 1, \cdots, n$, we determine the minimum cut $\left|\overline{V}_t\right| - 1$ times. The time complexity of finding a minimum cut (or maxflow) is $O\left(\left|\overline{V}_t\right|^2 \sqrt{\left|\overline{A}_t\right|}\right)$ by using Goldberg and Tarjan algorithm [23]. Therefore, the overall time complexity is $O\left(\left|\overline{V}_t\right|^3 \sqrt{\left|\overline{A}_t\right|}\right)$. This routine needs a considerably long computational time but always succeeds in finding the violated constraint.

In the separation routine based on the strongly connected components, we generate a network $\hat{G}_t\left(\hat{V}_t, \hat{A}_t\right)$, where $\hat{V}_t \equiv \left\{y_t^i, i \in I \middle| \overline{y}_t^i > 0\right\}$ and $\hat{A}_t \equiv \left\{e_{tt}^{ij}, i \neq j \in I \middle| (t,t) \in T^{ij}, \overline{e}_{tt}^{ij} > 0\right\}$. In the network, we could find some strongly connected components with more than two vertices. In each component S, we select the vertex $y_t^i \in S$ with the largest value $\overline{y}_t^i$; then, for S and $k = y_t^i$, we check if constraint (22) is violated by the current solution. If the constraint is variolated, we add it into current running formulation. The strongly connected components are found in a $O\left(\left|\hat{V}_t\right| + \left|\hat{A}_t\right|\right)$ operations by Tarjan's algorithm [24], and $O\left(\left|\hat{V}_t\right| + \left|\hat{A}_t\right|\right)$ operations are needed to check the violation. Therefore, the overall time complexity is $O\left(\left|\hat{A}_t\right|\right)$. This routine is significantly fast. However, it successfully determines the violated constraint only in the case of integer solution. More information about the separation routine based on the strongly connected components is available in [25].

4.2. Implementation

The formulation and the branch-and-cut procedure are implemented using the Gurobi LP/MIP solver 9.0.2 with Python 3.7. The separation routines are executed by the embedded callback

function. To find the minimum cut (or maxflow) and the strongly connected components, the efficient implementations in the Python library, Python-igraph 0.8.2, were used.

The following separation scenario was used in the computational experiments:

- For integral solutions, the separation routine based on the strongly connected components generates the violated inequalities for each time bucket;
- For fractional solution founded in i-th branch-and-bound nodes;
    - If $i \leq 1000$, the separation routine based on the strongly connected components generates the violated inequalities for each time bucket. In the absence of a violated inequality, the separation routine based on the minimum cut is performed for each time bucket.
    - If $1000 < i \leq 10,000$, the separation routine based on the strongly connected components generates the violated inequalities for each time bucket.
- If $i > 10,000$, no separation routine is performed.

5. Computational Experiments

This section describes three sets of computational experiments. The first is for comparison with MLOV-SM in [19], the CLSP with sequence dependent setup, setup carryover and "setup-splitting". The second is for the analysis of performance of M1 with test instances having long setup times. The third is for comparison with MM2 in [1], the CLSP with sequence dependent setup, setup carryover and "setup crossover".

Experiments were conducted on a Windows 2010 PC with 16 GB main memory, which was equipped with an Intel(R) Core(TM) i5-9500 @ 3.00 GHz CPU. Default parameter settings for Gurobi LP/MIP optimizer are used, except 600 s of time limit.

5.1. Test Instances

For the first and second tests, three groups of artificial test instances ("T", "L", "LA") were generated. The instances in "T" have short setup times and were designed to be used in the first test. The instances in "L", "LA" have long setup times and were designed to be used in the second test. They have the following features:

- Problem dimension: the problem dimension is set by (m, n), where m is the number of items and n is the number of time buckets. Therefore, each item is indexed as $i = 1, \cdots, m$. The following six problem dimensions were used: (10,10), (10,20), (10,30), (15,10), (15,20), (15,30).
- Setup time: for $i, j = 1, \cdots, m$
    - For group "T",
        - $\tau^{ij} = \frac{240}{m} \times \left|i - j\right| + 30 - m$, For $m = 10$, the minimum and maximum values of τ^{ij} are 44 and 236, respectively. For $m = 15$, the minimum and maximum values of τ^{ij} are 31 and 239.
    - For group "L",
        - $\tau^{ij} = \begin{cases} 24 \times \left|i - j\right| + 210 & \text{if } \left\lceil \frac{i}{5} \right\rceil = \left\lceil \frac{j}{5} \right\rceil \\ 24 \times \left|i - j\right| + 10 & \text{otherwise.} \end{cases}$, For $m = 10$, the minimum and maximum values of τ^{ij} are 34 and 426, respectively. Furthermore, items are divided into three groups: $\{1, 2, 3, 4\}$; $\{5, 6, 7, 8, 9\}$; and $\{10\}$, and most inter-group model changes have a longer setup change times than a bucket capacity of 240. For $m = 15$, the minimum and maximum values of τ^{ij} are 34 and 546, respectively. Moreover, items are divided by four groups: $\{1, 2, 3, 4\}$; $\{5, 6, 7, 8, 9\}$; $\{10, 11, 12, 13, 14\}$; and $\{15\}$, and most inter-group model changes have setup change times longer than 240.

- For group "LA",
  - $\tau^{ij} = \begin{cases} 24 \times |i-j| + 490 & \text{if } \left\lceil \frac{i}{5} \right\rceil = \left\lceil \frac{j}{5} \right\rceil \\ 24 \times |i-j| + 10 & \text{otherwise.} \end{cases}$ For $m = 10$, the minimum and maximum values of τ^{ij} are 34 and 706, respectively. Moreover, items are divided into three groups: $\{1,2,3,4\}$; $\{5,6,7,8,9\}$; and $\{10\}$, and an inter-group model change has setup change time longer than the sum of the capacity of two buckets, 480. For $m = 15$, the minimum and maximum values of τ^{ij} are 34 and 826, respectively. Moreover, items are divided into 4 groups: $\{1,2,3,4\}$; $\{5,6,7,8,9\}$; $\{10,11,12,13,14\}$; and $\{15\}$, and an inter-group model changes has setup change time longer than 480.

- Process time per unit:
  - $p_t^i = 1$, *for all* $i = 1, \cdots, m, t = 1, \cdots, n$.
- Capacity:
  - $C_t = \begin{cases} 240, & \text{for } t = 1, \cdots, n-1, \\ 2400, & \text{for } t = n. \end{cases}$ Enough capacity was given to the last bucket to prevent loss of sales.
- Costs:
  - $h_t^i = 3$, for all $i = 1, \cdots, m, t = 1, \cdots, n$.
  - $b_t^i = \begin{cases} 30, & \text{for all } i = 1, \cdots, m, t = 1, \cdots, n-2, \\ 300, & \text{for all } i = 1, \cdots, m, t \in \{n-1, n\}. \end{cases}$ Large backorder cost was given to bucket $n-1$ to minimize production in the last bucket.
  - $c^{ij} = \left\lceil \frac{\tau^{ij}}{10} \right\rceil$, for $i, j = 1, \cdots, m$.
- Demand: for problem dimension (m, n) For each group and problem dimension, initial demand set, named as (1), was generated manually according to the following criteria:
  - $\sum_{i \in I} \left| T^i \right| = 2n$
  - $\left| T^i \right| \geq \frac{2n}{m}$, for $i \in I$
  - All production can be finished before the last bucket.

To increase the reliability of the experiments, for each initial demand set, the due dates were randomly perturbed to generate four additional demand sets, named as (2), (3), (4), (5). The demands of an item having the same due date are merged into one.

For the last test, a test instance introduced in [1] was used. All the above test instances are available on [26].

5.2. First Test: Comparison with the Setup-Splitting Model (MLOV-SM)

As mentioned earlier, MLOV-SM is the model for the CLSP with sequence dependent setup, setup carryover and "setup-splitting". In addition, it considers situations in which multiple lots can be generated for an item within a time bucket. For direct comparison with our model (M1), MLOV-SM was modified and represented in Appendix A. The modified features are as follows:

- Setup operation from i to i is not allowed;
- At most one lot can be generated for an item within a time bucket;
- The initial setup state is not given but determined by solving the model.

M1 has $(2n-1)\times m^2+2m$ binary integer variables and $n\times\left(\sum_{i\in I}|T_i|+m+1\right)$ fractional variables, whereas MLOV-SM has nm^3+2nm^2+2nm binary variables and $3nm+2n$ real variables, much more than M1. Because the setup time of the data instance in the group "T" is less than the capacities of the buckets, we can directly compare the lower bounds of the two formulation. We compare the two formulations with respect to the size, tightness of the LP relaxation bound and solution quality. Table 2 shows the tightness of the LP relaxation bound of the formulations. The column headers of the table are given as follows:

- n: number of buckets;
- m: number of items;
- LB0: the objective value of LP relaxation without any additional cuts except GSECs;
- LB: the objective value of LP relaxation with cuts generated by Gurobi;
- BEST OBJ: the best integer solution value found within the time limit. Bold means optimal;
- Gap0: relative gap between LB0 and the best integer solution value (BEST OBJ) at the end.

$$GAP0 = 100\times\frac{BEST\ OBJ - LB0}{BEST\ OBJ}$$

- GAP0 improved: the difference in Gap0 between MLOV-SM and M1.

Table 2. The objective values of linear programming (LP)-relaxation of analysis of performance of (M1) and setup-splitting model (MLOV-SM).

m	*n*	Set	MLOV-SM			M1			GAP0 Improved	BEST OBJ
			LB0	LB	GAP0	LB0	LB	GAP0		
10	10	(1)	14	4285	100%	3612	4828	81%	19%	**19,431**
10	10	(2)	49	3529	100%	1655	3479	91%	9%	**17,970**
10	10	(3)	25	3852	100%	2390	3431	82%	18%	**13,144**
10	10	(4)	968	10,697	96%	9041	10,722	62%	33%	**24,102**
10	10	(5)	876	3699	89%	3174	3532	61%	28%	**8146**
10	20	(1)	406	4094	96%	2493	3898	77%	19%	**11,070**
10	20	(2)	557	14,482	98%	13,397	5939	60%	38%	**33,840**
10	20	(3)	1255	5359	90%	4555	5265	63%	27%	12,259
10	20	(4)	1630	6737	88%	5830	6789	57%	31%	13,463
10	20	(5)	933	4787	94%	4197	4775	73%	21%	15,343
10	30	(1)	2606	8134	84%	7387	8501	55%	29%	16,510
10	30	(2)	1102	5701	93%	4704	5904	69%	23%	15,388
10	30	(3)	4174	23,466	93%	20,615	26,529	64%	29%	57,589
10	30	(4)	12,678	47,233	88%	45,948	53,933	56%	32%	104,245
10	30	(5)	1040	7430	93%	6153	7318	61%	32%	15,975
15	10	(1)	23	3431	100%	1628	3054	89%	11%	**15,053**
15	10	(2)	139	5862	99%	4047	7767	76%	23%	**17,099**
15	10	(3)	27	1882	100%	1144	1870	87%	13%	**8647**
15	10	(4)	1157	3944	84%	3736	3993	49%	35%	**7305**
15	10	(5)	88	7749	100%	6350	7543	68%	31%	**20,002**
15	20	(1)	334	3630	97%	2715	3895	80%	18%	13,321
15	20	(2)	107	2663	99%	1716	2797	88%	11%	14,863
15	20	(3)	314	5697	99%	3415	6521	88%	11%	28,001
15	20	(4)	552	8882	98%	6761	9215	71%	27%	**23,141**
15	20	(5)	4384	26,926	92%	24,373	31,467	58%	35%	57,563
15	30	(1)	957	6001	94%	5173	6660	67%	27%	15,492
15	30	(2)	1384	7599	93%	6331	8478	66%	27%	18,469
15	30	(3)	1276	6561	92%	4505	7070	72%	20%	16,245
15	30	(4)	520	9044	98%	5531	10,631	75%	22%	22,412
15	30	(5)	938	5531	93%	4212	5865	68%	25%	13,060

In Table 2, we can find that M1 give much tight lower bound for the test instances.

Tables 3 and 4 represent the size and the test result of the formulations. The column headers of table are given as follows:

- #Vars: number of variables;
- #Cons: number of constraints (no subtour elimination constraints are counted here);
- #BVars: number of binary variables;
- GAP: relative gap between the best lower bound (Z_{LB}) and the best integer solution value (Z_I) at the end;

$$\text{Gap} = 100 \times \frac{Z_I - Z_{LB}}{Z_I}$$

- #B node: number of nodes visited in the branch-and-bound tree;
- Runtime: total running time;
- GSECs: number of generated subtour elimination constraints.

Tables 3 and 4 show that M1 has much compact sizes of LP relaxation and gives much better solutions then MLOV-SM.

Table 3. Test results of MLOV-SM for instance group T.

m	*n*	Set	#Vars	#Cons	#BVars	Gap	#B Node	Runtime
10	10	(1)	12,530	13,631	12,210	0%	11,544	54
10	10	(2)	12,530	13,631	12,210	0%	36,390	255
10	10	(3)	12,530	13,631	12,210	0%	26,189	154
10	10	(4)	12,530	13,631	12,210	0%	10,295	100
10	10	(5)	12,530	13,631	12,210	0%	5541	52
10	20	(1)	25,050	27,261	24,410	23%	31,564	600
10	20	(2)	25,050	27,261	24,410	0%	26,580	399
10	20	(3)	25,050	27,261	24,410	27%	25,209	600
10	20	(4)	25,050	27,261	24,410	31%	16,111	600
10	20	(5)	25,050	27,261	24,410	42%	11,777	600
10	30	(1)	37,570	40,891	36,610	27%	14,949	600
10	30	(2)	37,570	40,891	36,610	43%	13,386	600
10	30	(3)	37,570	40,891	36,610	42%	6217	600
10	30	(4)	37,570	40,891	36,610	29%	10,364	600
10	30	(5)	37,570	40,891	36,610	41%	11,469	600
15	10	(1)	39,035	41,431	38,565	38%	6169	600
15	10	(2)	39,035	41,431	38,565	12%	7393	600
15	10	(3)	39,035	41,431	38,565	29%	6431	600
15	10	(4)	39,035	41,431	38,565	18%	10,003	600
15	10	(5)	39,035	41,431	38,565	27%	6125	600
15	20	(1)	78,055	82,861	77,115	76%	2429	600
15	20	(2)	78,055	82,861	77,115	83%	1996	600
15	20	(3)	78,055	82,861	77,115	81%	1508	600
15	20	(4)	78,055	82,861	77,115	33%	4051	600
15	20	(5)	78,055	82,861	77,115	50%	1814	600
15	30	(1)	117,075	124,291	115,665	51%	2801	600
15	30	(2)	117,075	124,291	115,665	76%	1778	600
15	30	(3)	117,075	124,291	115,665	64%	2267	600
15	30	(4)	117,075	124,291	115,665	55%	1752	600
15	30	(5)	117,075	124,291	115,665	63%	2194	600

Table 4. Test results of M1 for instance group T.

m	*n*	Set	#Vars	#Cons	#BVars	Gap	#B Node	Runtime	GSECs
10	10	(1)	2370	639	2070	0%	2282	5	807
10	10	(2)	2350	617	2070	0%	10,117	26	1675
10	10	(3)	2360	628	2070	0%	22,125	36	2007
10	10	(4)	2360	628	2070	0%	8217	22	1468
10	10	(5)	2350	617	2070	0%	7327	20	1731
10	20	(1)	5240	1647	4270	0%	58,813	155	2323
10	20	(2)	5240	1647	4270	0%	8761	48	4003
10	20	(3)	5200	1605	4270	12%	213,454	600	4400
10	20	(4)	5280	1689	4270	29%	70,078	600	5317
10	20	(5)	5260	1668	4270	18%	132,385	600	3867
10	30	(1)	8440	2984	6470	17%	66,859	600	5603
10	30	(2)	8530	3077	6470	30%	58,134	600	6837
10	30	(3)	8500	3046	6470	14%	102,790	600	9691
10	30	(4)	8470	3015	6470	7%	123,380	600	5100
10	30	(5)	8530	3077	6470	28%	62,783	600	8550
15	10	(1)	4875	834	4530	0%	45,097	235	4429
15	10	(2)	4875	834	4530	0%	7003	46	2876
15	10	(3)	4875	834	4530	0%	28,266	93	3358
15	10	(4)	4875	834	4530	0%	38,934	145	3944
15	10	(5)	4875	834	4530	0%	21,151	92	3326
15	20	(1)	10,435	2084	9330	57%	29,770	600	9788
15	20	(2)	10,395	2042	9330	65%	30,017	600	9787
15	20	(3)	10,395	2042	9330	40%	20,828	600	9864
15	20	(4)	10,375	2021	9330	0%	46,690	305	8245
15	20	(5)	10,395	2042	9330	13%	75,191	600	6768
15	30	(1)	16,305	3641	14,130	36%	29,308	600	18,372
15	30	(2)	16,275	3610	14,130	30%	25,511	600	11,348
15	30	(3)	16,365	3703	14,130	35%	24,841	600	17,170
15	30	(4)	16,335	3672	14,130	15%	26,177	600	11,645
15	30	(5)	16,275	3610	14,130	36%	42,640	600	16,000

5.3. Second Test: For the Instance of Long Setup Times (L, LA)

The second test is to see if M1 is stable in solving problems with long setup time. Tables 5 and 6 show test results for L and LA, respectively. The gap value '-' in Table 5 indicates that no feasible solution found in the time limit.

Table 5. Test results of M1 for instance group L.

m	*n*	Set	#Vars	#Cons	#BVars	Gap	#B Node	Runtime	GSECs
10	10	(1)	2156	628	1866	0%	30,233	44	1052
10	10	(2)	2146	617	1866	0%	22,589	28	1038
10	10	(3)	2156	628	1866	0%	17,661	27	1005
10	10	(4)	2166	639	1866	0%	19,602	32	1024
10	10	(5)	2166	639	1866	0%	25,845	31	954
10	20	(1)	4836	1647	3866	0%	59,050	266	2108
10	20	(2)	4816	1626	3866	25%	174,286	600	2203
10	20	(3)	4816	1626	3866	12%	187,786	600	2106
10	20	(4)	4776	1584	3866	32%	108,165	600	2741
10	20	(5)	4816	1626	3866	6%	176,893	600	1650
10	30	(1)	7836	2984	5866	33%	40,977	600	3491
10	30	(2)	7836	2984	5866	30%	48,641	600	2755
10	30	(3)	7836	2984	5866	43%	29,950	600	4432
10	30	(4)	7746	2891	5866	36%	63,432	600	3510
10	30	(5)	7806	2953	5866	52%	30,385	600	4655

Table 5. *Cont.*

m	*n*	Set	#Vars	#Cons	#BVars	Gap	#B Node	Runtime	GSECs
15	10	(1)	4476	823	4141	0%	40,842	133	2120
15	10	(2)	4486	834	4141	0%	24,924	88	2090
15	10	(3)	4486	834	4141	0%	17,389	76	1865
15	10	(4)	4466	812	4141	0%	36,068	124	2408
15	10	(5)	4476	823	4141	0%	16,542	79	2336
15	20	(1)	9726	2063	8641	68%	15,187	600	5991
15	20	(2)	9686	2021	8641	70%	19,465	600	5356
15	20	(3)	9666	2000	8641	49%	28,431	600	4613
15	20	(4)	9726	2063	8641	64%	22,679	600	3930
15	20	(5)	9686	2021	8641	57%	26,496	600	3846
15	30	(1)	15,316	3641	13,141	98%	8968	601	9291
15	30	(2)	15,316	3641	13,141	–	8442	600	7097
15	30	(3)	15,286	3610	13,141	–	9727	601	5861
15	30	(4)	15,376	3703	13,141	80%	10,540	600	4914
15	30	(5)	15,316	3641	13,141	85%	7473	600	6544

Table 6. Test results of M1 for instance group LA.

m	*n*	Set	#Vars	#Cons	#BVars	Gap	#B Node	Runtime	GSECs
10	10	(1)	2094	628	1804	0%	10,212	15	745
10	10	(2)	2084	617	1804	0%	5779	13	929
10	10	(3)	2094	628	1804	0%	25,709	31	953
10	10	(4)	2104	639	1804	0%	3980	13	1105
10	10	(5)	2104	639	1804	0%	8797	17	833
10	20	(1)	4774	1647	3804	31%	135,029	600	1888
10	20	(2)	4754	1626	3804	23%	125,342	600	1960
10	20	(3)	4754	1626	3804	0%	48,764	165	1965
10	20	(4)	4714	1584	3804	30%	120,860	600	1990
10	20	(5)	4754	1626	3804	29%	118,298	600	1980
10	30	(1)	7774	2984	5804	54%	23,608	600	3127
10	30	(2)	7774	2984	5804	51%	51,117	600	2873
10	30	(3)	7774	2984	5804	64%	33,012	600	3085
10	30	(4)	7684	2891	5804	64%	33,178	600	3777
10	30	(5)	7744	2953	5804	55%	33,484	600	3640
15	10	(1)	4294	823	3959	0%	81,498	160	2164
15	10	(2)	4304	834	3959	0%	55,273	165	2116
15	10	(3)	4304	834	3959	0%	106,169	249	2128
15	10	(4)	4284	812	3959	0%	127,983	256	2905
15	10	(5)	4294	823	3959	0%	220,737	344	1769
15	20	(1)	9544	2063	8459	66%	35,127	600	3679
15	20	(2)	9504	2021	8459	62%	27,643	600	3878
15	20	(3)	9484	2000	8459	58%	35,291	600	3297
15	20	(4)	9544	2063	8459	72%	21,749	600	3921
15	20	(5)	9504	2021	8459	70%	34,165	600	4398
15	30	(1)	15,134	3641	12,959	88%	9876	601	5509
15	30	(2)	15,134	3641	12,959	91%	9416	600	5512
15	30	(3)	15,104	3610	12,959	80%	10,503	600	5484
15	30	(4)	15,194	3703	12,959	90%	10,928	600	5329
15	30	(5)	15,134	3641	12,959	92%	7755	600	5304

Tables 4–6 show that M1 reliably optimizes the problem instances with ten buckets and give relatively good feasible solution for the problem instances with twenty buckets.

5.4. Third Test: Comparison with the Setup Crossover Model (MM2)

Ramya, et al. [1] introduced the only research result on CLSP with sequence-dependent setup, setup carryover, and setup crossover. In the model, no idle times are allowed during setup operation and setup carryover. MM2 is a formulation introduced in [1] for the problem. But MM2 is too big to solve even problems of moderate size. MM2 has $8mn^3 + 4m^2n^2 + 6mn^2 + 4m^2n + 6mn + n$ binary integer variables and more than 150 types of constraints. Ramya, et al. [1] reported that it took more than 26,000 s to solve a test instance with 15 items and 10 buckets and that the solver failed with a file error after 8.5 h due to insufficient space in the hard disk while solving a test instance with 15 items and 15 buckets.

For the comparison with MM2, a modified version of M1 was developed and used for the test. The modified formulation, referred to as M2, needs two types of new variables described in Table 7.

Table 7. New variables for M2.

w_t^i	Variable indicating cleanup state: 1 if production of item i is completed and model change is prepared in bucket t, and 0 otherwise.
$\bar{z}_t^i$	Variable indicating carryover of completed job: 1 if the setup state is for completed item i at the beginning of bucket t, and 0 otherwise

With the variables, M2 is defined as follows:
Min (3) subject to (4)–(7), (9) and (11)–(19) and

$$\sum_{i\in I}\left(z_t^i + \bar{z}_t^i\right) + \sum_{i,j\in I|i\neq j}\ \sum_{(s,u)\in T^{ij}|s<t,u\geq t} e_{su}^{ij} \leq 1, \text{ for all } t = 1,\cdots,n+1 \tag{27}$$

$$z_{t+1}^i + w_t^i = y_t^i, \text{ for all } i \in I, t = 1,\cdots,n \tag{28}$$

$$\bar{z}_{t+1}^i + \sum_{j\in I|i\neq j}\ \sum_{(t,u)\in T^{ij}} e_{tu}^{ij} = w_t^i + \bar{z}_t^i, \text{ for all } i \in I, t = 1,\cdots,n \tag{29}$$

$$L_t \leq C_t \cdot \sum_{i\in I}\left(w_t^i + \bar{z}_t^i\right), \text{ for all } t = 1,\cdots,n \tag{30}$$

$$\bar{z}_t^i \in \{0,1\} \text{ for all } i \in I, t = 1,\cdots,n+1 \tag{31}$$

$$y_t^i \in \{0,1\} \text{ for all } i \in I, t \in T \tag{32}$$

$$\bar{z}_1^0 = 1. \tag{33}$$

Here, 0 in $\bar{z}_1^0$ means the initial setup item. M2 has $(2n-1)\times m^2 + 2nm + 3m$ binary integer variables, whereas MM2 has $8mn^3 + 4m^2n^2 + 6mn^2 + 4m^2n + 6mn + n$ binary variables. We can say that M2 is a much compact formulation than MM2.

Ramya et al. [1] used six problem instances for the test, but just one instance with 10 items and 15 buckets was reported, so that we can use only that. They reported that MM2 found an optimal solution of the problem in 425 s. M2 was tested with the instance and get the result reported in Table 8. M2 solved the problem just in 2 s to the optimality. In addition, the optimal solution provided the same production plan as [1], except when idle time occurred. Even considering the performance differences in the computers used and limited test instance, it is judged that these differences show the superiority of M2.

Table 8. Test results of M2.

m	*n*	#Vars	#Cons	#BVars	Gap	#B Node	Runtime	GSECs
10	15	3752	900	3238	0%	1	1.53	44

6. Conclusions

In this paper, the CLSP with sequence dependent setup, setup carryover, and setup crossover was considered. A new mixed integer programming formulation was introduced. The formulation is based on three building blocks: the facility location extended formulation [2]; the setup variables with indices for the starting and the completion time periods [14]; and exponential number of generalized subtour elimination constraints (GSECs) [18]. A modified version of the separation routine of [25] was adopted to generate the violated GSECs.

Three groups of artificial test instances and an instance from [1] were used for computational experiments. The computational results showed that the proposed formulation outperformed the models from the literature, but the formulation has still large number of binary setup variables, $O\left(n \times |I|^2\right)$, to solve the problems with more than twenty buckets in 10 min.

To overcome this drawback, studies to reduce the number of setup variables are needed. These studies could involve using the column generation technique or heuristics based on variable fixing.

Funding: This study was supported by research fund from Chosun University, 2017.

Conflicts of Interest: The author declares no conflict of interest.

Appendix A. Modified MLOV-SM Model

The formulation for the modified MLOV-SM model [19] is follows:

$$\textbf{Min} \sum_{i,j\in I}\sum_{t=1}^{n} c_{ij}y_{ijt} + \sum_{i\in I}\sum_{t=1}^{n} h_t^i I_{it} + \sum_{i\in I}\sum_{t=1}^{n} b_t^i B_{it} \tag{A1}$$

Subject to

$$I_{jt-1} - B_{jt-1} + x_{ji} - I_{jt} + B_{jt} = d_t^j, \text{ for all } j \in I, t = 1, \cdots, n \tag{A2}$$

$$\sum_{i\in I} p_t^i x_{it} + \sum_{i,j\in I} \tau_{ij} y_{ijt} + S_{t-1} - S_t + slk_t = C_t, \text{ for all } t = 1, \cdots, n \tag{A3}$$

$$x_{jt} \le \frac{C_t}{p_t^i} \times z_{jt}, \text{ for all } j \in I, t = 1, \cdots, n \tag{A4}$$

$$\sum_{i\in I} \alpha_{it} = 1, \text{ for all } t = 1, \cdots, n+1 \tag{A5}$$

$$\alpha_{it} + \sum_{j\in I}\left(y_{jit} - OLS_{jit}\right) = z_{it}, \text{ for all } i \in I, t = 1, \cdots, n \tag{A6}$$

$$\alpha_{it+1} + \sum_{j\in I} y_{ijt} - \sum_{j\in I} OLS_{jit} = z_{it}, \text{ for all } i \in I, t = 1, \cdots, n \tag{A7}$$

$$OLS_{ijt} + a_{ijt}^k \le y_{ijt}, \text{ for all } i, j, k \in I, t = 1, \cdots, n \tag{A8}$$

$$\alpha_{kt} + \sum_{i\in I} a_{ikt}^k = z_{it}, \text{ for all } k \in I, t = 1, \cdots, n \tag{A9}$$

$$\alpha_{it} + \sum_{j\in I} a_{jit}^k \ge \sum_{j\in I} a_{ijt}^k, \text{ for all } i \ne k \in I, t = 1, \cdots, n \tag{A10}$$

$$S_t \le \sum_{i,j\in I} \tau^{ij} OLS_{ijt}, \text{ for all } t = 1, \cdots, n \tag{A11}$$

$$\sum_{j\in I} OLS_{jit} \le \alpha_{i,t+1}, \text{ for all } i \in I, t = 1, \cdots, n \tag{A12}$$

$$OLS_{ijt} \leq y_{ijt}, \text{ for all } i, j \in I, t = 1, \cdots, n \tag{A13}$$

$$x_{jt}, I_{jt}, B_{jt} \geq 0, \text{ for all } j \in I, t = 1, \cdots, n \tag{A14}$$

$$S_t, slk_t \geq 0, \text{ for all } t = 1, \cdots, n \tag{A15}$$

$$z_{it}, y_{ijt}, OLS_{ijt}, a^k_{ijt} \in \{0, 1\}, \text{ for all } i, j, k \in I, t = 1, \cdots, n \tag{A16}$$

$$\alpha_{it} \in \{0, 1\}, \text{ for all } i \in I, t = 1, \cdots, n+1 \tag{A17}$$

$$I_{in}, B_{in}, \ I_{i0}, B_{i0} = 0, \text{ for all } i \in I \tag{A18}$$

$$a^k_{kjt} = 0, \text{ for all } j, k \in I, t = 1, \cdots, n \tag{A19}$$

$$S_0 = 0 \tag{A20}$$

$$y_{jjt} = 0, \text{ for all } j \in I, t = 1, \cdots, n \tag{A21}$$

References

1. Ramya, R.; Rajendran, C.; Ziegler, H.; Mohapatra, S.; Ganesh, K. *Capacitated Lot Sizing Problems in Process. Industries*; Springer: Cham, Switzerland, 2019.
2. Pochet, Y.; Wolsey, L.A. *Production Planning by Mixed Integer Programming*; Wiley: New York, NY, USA, 2006.
3. Salomon, M.; Kroon, L.G.; Kuik, R.; Wassenhove, L.N. Some extensions of the discrete lot sizing and scheduling problem. *Manag. Sci.* **1991**, *37*, 801–812. [CrossRef]
4. Sox, S.R.; Gao, Y. The capacitated lot sizing with setup carry-over. *IIE Trans.* **1999**, *31*, 173–181. [CrossRef]
5. Suerie, C.; Stadtler, H. The capacitated lot-sizing problem with linked lot sizes. *Manag. Sci.* **2003**, *49*, 1039–1054. [CrossRef]
6. Menezes, A.A.; Clark, A.; Almada-Lobo, B. Capacitated lot-sizing and scheduling with sequence-dependent, period-overlapping and non-triangular setups. *J. Sched.* **2011**, *14*, 209–219. [CrossRef]
7. Ramya, R.; Rajendran, C.; Ziegler, H. Capacitated lot-sizing problem with production carry-over and set-up splitting: Mathematical models. *Int. J. Prod. Res.* **2016**, *54*, 2332–2344. [CrossRef]
8. Dantzig, G.; Fulkerson, R.; Johnson, S. Solution of a large-scale traveling-salesman problem. *Oper. Res.* **1954**, *2*, 393–410. [CrossRef]
9. Kopanos, G.M.; Puigjaner, L.; Maravelias, C.T. Production planning and scheduling of parallel continuous processes with product families. *Ind. Eng. Chem. Res.* **2011**, *50*, 1369–1378. [CrossRef]
10. Mohan, S.; Gopalakrishnan, M.; Marathe, R.; Rajan, A. A note on modelling the capacitated lot-sizing problem with set-up carryover and set-up splitting. *Int. J. Prod. Res.* **2012**, *50*, 5538–5543. [CrossRef]
11. Camargo, V.C.B.; Toledo, F.M.B.; Almada-Lobo, B. Three time-based scale formulations for the tow-stage lot sizing and scheduling in process industries. *J. Oper. Res. Soc.* **2012**, *63*, 1613–1630. [CrossRef]
12. Fiorotto, D.J.; Jans, R.; De Araujo, S.A. An analysis of formulations for the capacitated lot sizing problem with setup crossover. *Comput. Ind. Eng.* **2017**, *106*, 338–350. [CrossRef]
13. Sung, C.; Maravelias, C.T. A mixed-integer programming formulation for the general capacitated lot-sizing problem. *Comput. Chem. Eng.* **2008**, *32*, 244–259. [CrossRef]
14. Belo-Filho, M.A.F.; Almada-Lobo, B.; Toledo, F.M.B. Models for capacitated lot-sizing problem with backlogging, Setup carryover and crossover. *J. Oper. Res. Soc.* **2014**, *65*, 1735–1747. [CrossRef]
15. Alamda-Lobo, B.; Klabjan, D.; Carravilla, M.A.; Oliveira, J.F. Single machine multi-product capacitated lot sizing with sequence-dependent setups. *Int. J. Prod. Res.* **2007**, *45*, 4873–4894. [CrossRef]
16. Clark, A.; Morabito, R.; Toso, E. Production setup-sequencing and lot-sizing at an animal nutrition plant through ATSP subtour elimination and patching. *J. Sched.* **2010**, *13*, 111–121. [CrossRef]
17. Clark, A.; Mahdieh, M.; Rangel, S. Production lot sizing and scheduling with non-triangular sequence-dependent setup times. *Int. J. Prod. Res.* **2014**, *52*, 2490–2503. [CrossRef]
18. Taccari, L. Integer programming formulations for the elementary shortest path problem. *Eur. J. Oper. Res.* **2016**, *252*, 122–130. [CrossRef]

19. Mahdieh, M.; Clark, A.; Bijari, M. A novel flexible model for lot sizing and scheduling with non-triangular, period overlapping and carryover setups in different machine configurations. *Flex. Serv. Manuf. J.* **2018**, *30*, 884–923. [CrossRef]
20. Yang, L.; Zhang, C.; Jian, J.; Meng, K.; Xu, Y.; Dong, Z. A novel projected two-binary-variable formulation for unit commitment in power systems. *Appl. Energy* **2017**, *187*, 732–745. [CrossRef]
21. Nemhauser, G.L.; Wolsey, L.A. *Integer and Combinatorial Optimization*; Wiley: New York, NY, USA, 1988.
22. Balas, E. The prize collecting traveling salesman problem. *Networks* **1989**, *19*, 621–636. [CrossRef]
23. Goldberg, A.V.; Tarjan, R.E. A new approach to the maximum-flow problem. *J. ACM* **1988**, *35*, 921–940. [CrossRef]
24. Tarjan, R. Depth-first search and linear graph algorithms. *SIAM J. Comput.* **1972**, *1*, 146–160. [CrossRef]
25. Drexl, M. A note on the separation of subtour elimination constraints in elementary shortest path problems. *Eur. J. Oper. Res.* **2013**, *229*, 595–598. [CrossRef]
26. CLSP Data Set. Available online: http://ie1.chosun.ac.kr/~{}jhkang/data/CLSP.zip (accessed on 26 June 2020).