IISE Transactions

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/uiie21

A Lot-Sizing Problem in Deliberated and Controlled Co-Production Systems

Bahadır Pamuk, Semra Ağralı, Z. Caner Taşkın & Banu Kabakulak

To cite this article: Bahadır Pamuk, Semra Ağralı, Z. Caner Taşkın & Banu Kabakulak (2021): A Lot-Sizing Problem in Deliberated and Controlled Co-Production Systems, IISE Transactions, DOI: 10.1080/24725854.2021.2022250

To link to this article: https://doi.org/10.1080/24725854.2021.2022250

A Lot-Sizing Problem in Deliberated and Controlled Co-Production Systems

Bahadır Pamuk[a], Semra Ağralı*[b], Z. Caner Taşkın[a], Banu Kabakulak[c]

[a]Department of Industrial Engineering, Boğaziçi University, İstanbul, Turkey

[b]Department of Industrial Engineering, MEF University, İstanbul, Turkey

[c]Department of Industrial Engineering, İstanbul Bilgi University, İstanbul, Turkey

*Corresponding author semra.agrali@gmail.com

Abstract

We consider an uncapacitated lot sizing problem in co-production systems, in which it is possible to produce multiple items simultaneously in a single production run. Each product has a deterministic demand to be satisfied on time. The decision is to choose which items to co-produce and the amount of production throughout a predetermined planning horizon. We show that the lot sizing problem with co-production is strongly NP-Hard. Then, we develop various mixed-integer linear programming (MILP) formulation of the problem and show that LP relaxations of all MILPs are equal. We develop a separation algorithm based on a set of valid inequalities, lower bounds based on a dynamic lot-sizing relaxation of our problem and a constructive heuristic that is used to obtain an initial solution for the solver, which form the basis of our proposed Branch & Cut algorithm for the problem. We test our models and algorithms on different data sets and provide the results.

Keywords: Co-production; Lot-Sizing; MILP; Branch & Cut.

1 Introduction

Co-production is a process where several different products are produced simultaneously in the same production run. Co-production either occurs because of physical or chemical nature of the production system, or because the system is designed to produce multiple products simultaneously in order to effectively use scarce resources. Co-produced units may only differ in quality as in semi-conductor production (Bitran and Gilbert, 1994; Bitran and Dasu, 1992) or may be completely different products as in float glass production (Öner and Bilgiç, 2008; Taşkın and Ünal, 2009). As a development strategy, concept of circular economy aims to maintain rapid economic growth while considering scarcity of raw materials (Yuan et al., 2006). There is a strong relation between circular economy and co-production as circular economy aims to minimize the waste (or use) of raw materials while producing goods or services (Brahimi et al., 2017; Suzanne et al., 2020). In this sense co-production serves to achieve this goal and it is very important in today's manufacturing industry.

We can analyze co-production in different categories based on the nature of the production: (i) deliberated or non-deliberated; (ii) controlled or uncontrolled (Suzanne et al., 2020). If it is possible to control the parameters of the production, such as the products that can be produced as co-products and their quantities, then the system has *controlled* co-production. On the other hand, if it is not possible to control the types or quantities of products when the production run starts or it is prohibitively expensive to control the system, then the system has *uncontrolled* co-production. Moreover, if it is possible to manufacture each product either by itself or by means of co-production, and co-production is desirable in terms of cost and production, then the system has *deliberated* co-production. For example, in metal press shops often multiple products are produced simultaneously from a single metal sheet, and hence the co-production is deliberated and controlled. In contrast, semi-conductor and float glass production are neither deliberated nor controlled (Bitran and Gilbert, 1994; Taşkın and Ünal, 2009). In this study, we focus on lot-sizing problems occurring in deliberated and controlled co-production systems.

As an example, we may consider a case that occurs in the plastics industry. Figure 1 depicts a plastic mould that can simultaneously produce different plastics products. As seen from the figure, the fixtures are changeable, and with a new fixture, the set of co-produced products can be adjusted. Since it is possible to produce different shapes of plastics by changing the fixtures and with each fixture we know the quantities and types of products that are co-produced, the system is controlled. Furthermore, since it is possible to use the mould to either produce a single product type or to co-produce different products, the system is deliberated.

From an applications point of view, our problem can be found at production systems in which multiple products, often small parts, can be produced by fitting multiple parts into a single die or machine slot. Although we focus on co-production, one may encounter a similar problem in supply chain systems (Stern, 2006): Consider a company that needs to buy a certain amount of supplies (material A, B and C), and there exist suppliers (supplier 1 and 2) that offer a set of deals for different combinations of materials. For example supplier 1 provides (500 kg of material A) + (500 units of material B) for \$X while supplier 2 offers (1000 kg of material A) + (100 units of material B) + (100 kg of material C) for \$Y. This results in deciding on the suppliers to work with and the amount of each bundle to buy from these selected suppliers, which is similar to a production planning problem in a co-production system.

Lot sizing problems are well studied in the literature. Since Wagner and Whitin (1958) published their seminal paper, substantial research has been done in the area. Hence, various lot sizing problems have been studied. For an extensive review, we refer the reader to Brahimi et al. (2006) and its updated version (Brahimi et al., 2017). Since the focus of our study is lot-sizing in co-production systems, we discuss the corresponding literature rather than general lot-sizing literature.

A type of lot-sizing problem appears as coordinated lot sizing problem, where the products are produced as families. In coordinated lot sizing problems, a family of products shares a fixed setup cost (Robinson et al., 2009). In addition to the fixed cost, a minor setup cost also exists for individual products inside a product family if they are decided to be produced. Variable production cost is similar to that of dynamic lot sizing problems. Despite having a shared fixed cost for multiple products as families, it does not capture the notion of co-production since every product inside the same family is not necessarily produced simultaneously. In co-production it is usually not possible to group products as families and co-produced items are produced simultaneously without having minor set up costs.

Bitran and Dasu (1992) and Bitran and Gilbert (1994) focus on co-production with random yields in semiconductor production. In their context, it is possible to substitute a lower tier product with a higher tier one. This is called serially nested co-production. The problem is divided into two sub-problems as “morning problem,” in which production decisions are made, and “afternoon problem,” in which products are allocated to customers after yields are known. In Bitran and Dasu (1992), the objective is to maximize expected profit whereas in Bitran and Gilbert (1994) it is the minimization of expected cost comprised of production, inventory holding, and shortage costs.

Öner and Bilgiç (2008) study uncontrolled co-production in float glass manufacturing with constant holding cost rate and fixed sequence independent setup costs where the substitution of products is not allowed. They develop a continuous time economic lot scheduling model to find a common cycle schedule. Taşkın and Ünal (2009) also study co-production in float glass manufacturing focusing on tactical level planning. They develop a mathematical model that is highly specialized for that purpose. Vidal-Carreras et al. (2012) study deliberated and controlled co-production with non-substitutable demand. Similar to Öner and Bilgiç (2008), their model is a continuous economic lot scheduling problem with the aim of finding a common cycle time. Costs are constant, fixed and sequence independent, and only two products are considered. Rafiei et al. (2015) consider co-production with sequence dependent setup times and demand uncertainty. There are production families, and recipes in the same production family require no changeover cost. They provide a case study on demand driven wood re-manufacturing mills, and propose a three step methodology to solve wood re-manufacturing industrial problem.

To the best of our knowledge there is a single lot-sizing paper, Ağralı (2012), that considers a setting where co-production exists. In that study co-production is controlled but not deliberated, i.e., there is only a single set of products to be co-produced as a single co-production unit. The problem is to decide on when and how much to produce that specific set of products. It is shown that well-known zero-inventory policy holds for at least one of the products, and a dynamic programming (DP) recursion is given that runs in polynomial time. Although Ağralı (2012)'s paper is the closest work to our paper, there are some major differences between the two studies. In particular, we consider deliberated co-production where the decision maker has the option of co-produce or not to co-produce. Moreover, there is no single set of products, and the products may appear in different co-production units to be produced as co-products. In our case, zero inventory property does not hold, and hence, none of the solution methods proposed in the literature is directly applicable to our problem.

Our contribution to the lot sizing literature can be summarized as follows:

i) We formally define a deliberated co-production problem for the first time in the dynamic deterministic lot sizing literature.
ii) We investigate the computational complexity of the problem, and prove that it is NP-Hard in the strong sense.
iii) We develop alternative mixed-integer programming formulations for the problem and investigate their strengths in terms of the tightness of their linear programming relaxations.
iv) We propose a valid inequality, an efficient separation algorithm and a series of lower bounds, which we combine in a Branch & Cut algorithm. We also design a heuristic method to find high-quality feasible solutions to the problem.

Our experiments show that our proposed methods are individually very effective, and they significantly increase the number of best solutions obtained when CPLEX cuts are on.

The remainder of the paper is as follows: Section 2 gives the formal problem definition and the proposed mathematical model with NP-Hardness proof. Section 3 provides alternative mixed-integer linear programs that we developed for our problem and the equivalence of all formulations in terms of LP relaxations. We provide a separation algorithm based on valid inequalities that we propose, a heuristic method that is used as an initial solution for the solver and improvement procedures that are based on a DLSP relaxation to be used in our Branch & Cut algorithm in Section 4. We give our computational experiments in Section 5, and finally conclude the paper with a brief summary and future work in Section 6.

2 Problem Description and Mathematical Model

We consider a dynamic uncapacitated lot-sizing problem (DLSP) in a production system where co-production exists. There is a set of products, *J*, where products can be co-produced in different combinations. We call a possible combination of products that can be co-produced together a "co-production unit (CU)," and denote these CUs as set *I*. We model time as a finite sequence of discrete time points, indexed as $t \in T$, as in DLSP.

There are finitely many CUs indexed by $i \in I$, and each produces a finite set of predefined products, *J*(*i*). When one CU of type *i* is to be produced, all products included in set *J*(*i*) are co-produced with certain production ratios, α_i^j , i.e., when one CU *i* is produced, then α_i^j units of product *j* are produced. Producing a CU requires a deterministic time-dependent fixed cost, f_t^i , and a variable cost, c_t^i . Each product $j \in J$ has dynamic deterministic demand, d_t^j , and incurs a holding cost, h_t^j , for each item carried in inventory. Backlogging is not allowed. The aim is to find a production plan with minimum possible cost that satisfies all demand on time. We provide a list of symbols that we use in modeling in Table 1.

There may be several ways to produce each product, i.e., a product may appear in multiple CUs. It is also possible for a CU to include only one product, which allows us to model deliberated co-production. Figure 2 depicts a production system that produces products A, B and C from a metal sheet, where there are six possible CUs defined. *CU1* produces A and C with $\alpha_1^A = \alpha_1^C = 2$. *CU2*, *CU4* and *CU6* produce single products where $\alpha_2^C = 4, \alpha_4^B = 2$, and $\alpha_6^A = 4$. *CU3* produces C and B with $\alpha_3^B = 1$ and $\alpha_3^C = 2$. Finally, *CU5* produces B and A with $\alpha_5^A = 2$ and $\alpha_5^B = 1$.

Note that, by definition of α_i^j, CUs include physical production restrictions if there are any. For example, CUs given in Figure 2 include a restriction of not producing A, B and C together. We refer to this type of system as Deliberated and Controlled Co-Production (DCCP).

Let decision variable x_t^i denote production amount of CU *i* in period *t*, and let binary variable y_t^i take value 1 if and only if production of CU *i* takes place in period *t*. Moreover, let s_t^j variable denote the inventory of product *j* at the end of period *t*. Then, a mixed-integer linear programming model (MILP) for DCCP can be written as follows:

$$\textbf{IP1}: \quad \text{Minimize} \quad \sum_{t \in T} \Big(\sum_{i \in I} \big(f_t^i y_t^i + c_t^i x_t^i \big) + \sum_{j \in J} h_t^j s_t^j \Big) \qquad (1)$$

$$\text{Subject to} \quad s_{t-1}^j + \sum_{i \in I} \alpha_i^j x_t^i - s_t^j = d_t^j, \quad \forall t \in T,\ j \in J; \qquad (2)$$

$$x_t^i \le \max_{j \in J(i)} \Big\{ \frac{d_{tT}}{\alpha_i^j} \Big\} y_t^i, \quad \forall t \in T, i \in I; \qquad (3)$$

$$x_t^i \ge 0, \quad \forall t \in T, i \in I; \qquad (4)$$

$$s_t^j \ge 0, \quad \forall t \in T,\ j \in J; \qquad (5)$$

$$y_t^i \in \{0,1\}, \quad \forall t \in T, i \in I. \qquad (6)$$

Objective function (1) minimizes the total fixed and variable costs of production and holding cost of products over the planning horizon. Constraints (2) are the inventory flow balance constraints. Constraints (3) enforce that if a production occurs, i.e., $x_t^i > 0$, then binary set up variables take the value of 1, i.e., $y_t^i = 1$. Constraints (4)–(6) are non-negativity and binary restrictions, respectively.

We first show that the optimization problem given as IP1 is strongly NP-Hard. In order to prove this, we first define the decision version of our optimization problem, D-DCCP.

D-DCCP. Given a set of CUs *I*, each with fixed cost, f_t^i, and variable cost, c_t^i; a set of products, *J*, each with demand, d_t^j, and holding cost, h_t^j; production ratios, α_i^j; and a positive integer, *K*, does there exist a feasible production plan (such that all demands are satisfied on time), having total cost no more than *K*?

Proposition 1. D-DCCP is strongly NP-Complete.

Proof. Given an instance of sets *I*, *J*, *T* and data $\alpha_i^j, f_t^i, c_t^i, d_t^j, h_t^j$, a positive number *K*, and a "guess" x_t^i; computing $y_t^i = 1$ if $x_t^i > 0$ and $y_t^i = 0$ otherwise; $s_t^j = \sum_{k \le t} \sum_{i \in I} (x_k^i \alpha_i^j - d_k^j)$ and computing if $\sum_{t \in T} \left(\sum_{i \in I} (f_t^i y_t^i + c_t^i x_t^i) + \sum_{j \in J} h_t^j s_t^j \right) \le K$ can be done in polynomial time. Hence, DCCP is in NP.

To show that D-DCCP is NP-complete, consider the following strongly NP-complete problem (Garey and Johnson, 1979).

MINIMUM COVER PROBLEM (MCP). Given a collection *C* of subsets of a finite set *S*, positive integer $K \le |C|$, does *C* contain a cover for *S* of size *K* or less, i.e., a subset $C' \subseteq C$ with $|C'| \le K$ such that every element of *S* belongs to at least one member of C'?

An arbitrary instance of MCP can be reduced to a particular instance of D-DCCP as follows: Let *S* in MCP correspond to set *J* in D-DCCP instance, and every subset in set *C* be indexed by *i* such that $C = \{C_i\}$. Let $|T| = 1,\ f_1^i = 1,\ c_1^i = 0,\ d_1^j = 1, \forall j \in J \text{ and } \forall i \in I$. Now for each $i \in I$ and $j \in J$ define subsets $J_i \in J$ for each $i \in I$ as $J_i = \{ j \mid j : \alpha_i^j = 1\}$, and define α_i^j to be equal to 1 if $j \in C_i$, and 0, otherwise. Observe that this transformation can easily be performed in polynomial time and the size of the transformed data is polynomially bounded.

In order to complete the proof, we need to show that an arbitrary instance of MCP is a yes-instance if and only if the transformed D-DCCP instance is a yes-instance.

Suppose that MCP instance is a yes-instance. Then, there exists a subset $C' \subseteq C$ such that $|C'| \leq K$ and $\bigcup C' = S$. Since selecting $C' \subseteq C$ in MCP corresponds to selecting $I' \subseteq I$ in D-DCCP, $\sum_{i \in I} y_t^i = |I'| = |C'| \leq K$.

Selecting $\alpha_i^j \in \{0,1\}$ and corresponding subsets J_i as described previously will ensure $\bigcup_{i \in I'} J_i = J$ since $\bigcup C' = S$,

which means each product $j \in J$ will be produced at least once by selecting $I' \subseteq I$ for production. Since $d_1^j = 1$ for all $j \in J$, production plan feasibility is ensured by $\bigcup_{i \in I'} J_i = J$. Thus, the D-DCCP instance is a yes-instance.

Now, suppose that MCP instance is a no-instance. Then, it is not possible to select $I' \subseteq I$ such that $|I'| \leq K$ and $\bigcup_{i \in I'} J_i = J$. The only way to satisfy production feasibility is to select a subset of *I* as I' such that all products $j \in J$ is included in these subsets at least once, i.e., $\bigcup_{i \in I'} J_i = J$, where $|I'| \leq K$. However, this is not possible by the assumption that MCP instance is a no-instance. Hence, the D-DCCP instance is a no-instance. □

3 Alternative MIP Formulations and Their Properties

In this section, we develop three alternative formulations to IP1 and show the equivalence of these in terms of their LP relaxations.

3.1 Alternative Formulations

We can reduce the number of variables of IP1 by representing inventory variables, s_t^j, in terms of production variables, x_t^i, and demand, d_t^j, i.e., $s_t^j = \sum_{k=1}^{t} \left(\sum_{i \in I(j)} x_k^i \alpha_i^j - d_k^j \right)$. We call the resulting formulation as IP2.

$$\textbf{IP2}: \quad \text{minimize} \quad \sum_{t \in T} \sum_{i \in I} \left(f_t^i y_t^i + a_t^i x_t^i \right) - H \qquad (7)$$

$$\text{subject to} \quad \sum_{k=1}^{t} \sum_{i \in I(j)} \alpha_i^j x_k^i \geq d_{1t}^j, \qquad \forall t \in T,\ j \in J \qquad (8)$$

$$x_t^i \leq \max_{j \in J(i)} \left\{ \frac{d_{tT}}{\alpha_i^j} \right\} y_t^i, \qquad \forall t \in T, i \in I \qquad (9)$$

$$x_t^i \geq 0, \qquad \forall t \in T, i \in I \qquad (10)$$

$$y_t^i \in \{0,1\}, \qquad \forall t \in T, i \in I. \qquad (11)$$

where $a_t^i = c_t^i + \sum_{j \in J(i)} \alpha_i^j h_{tT}^j$ and $H = \sum_{t \in T} \sum_{j \in J} h_t^j d_{1t}^j$.

We note that IP2 formulation may improve solution times since it has fewer number of variables compared to IP1 formulation. However, the constraint matrix is denser than IP1, which may create computational difficulties.

The simple plant location formulation of DLSP is given by Wagelmans, Van Hoesel, and Kolen (Wagelmans et al.). They show that this formulation gives the convex hull of DLSP based on the results in Krarup and Bilde (1977). We develop another MIP formulation based on the simple plant location formulation of DLSP, in which production variables are disaggregated in terms of periods where produced items are consumed by demand. Let $\Theta_{tt'}^{j}$ continuous variables represent the production amount of product $j \in J$, that is produced in period $t \in T$ to be consumed in period $t' \in T, t' \geq t$. However, unlike the simple plant location formulation, eliminating x_t^i variables does not appear to be possible since production costs depend on the amount of CUs produced, not products. Therefore, constraints (14) are needed to relate $\Theta_{tt'}^{j}$ variables to x_t^i variables, and the demand satisfaction constraint is revised as in Equation (13). Other constraints and the objective function remain the same as those of IP2:

$$\textbf{IP3}: \quad \text{minimize} \quad \sum_{t \in T} \sum_{i \in I} \left(f_t^i y_t^i + a_t^i x_t^i \right) - H \qquad (12)$$

$$\text{subject to} \quad \sum_{t \leq t'} \Theta_{tt'}^{j} = d_{t'}^{j}, \quad \forall t' \in T, j \in J \qquad (13)$$

$$\sum_{t' \geq t} \Theta_{tt'}^{j} \leq \sum_{i \in I(j)} \alpha_i^j x_t^i, \quad \forall t \in T, j \in J \qquad (14)$$

$$x_t^i \leq \max_{j \in J(i)} \left\{ \frac{d_{tT}}{\alpha_i^j} \right\} y_t^i, \quad \forall t \in T, i \in I \qquad (15)$$

$$x_t^i \geq 0, \quad \forall t \in T, i \in I \qquad (16)$$

$$\Theta_{tt'}^{j} \geq 0, \quad \forall t, t' \in T, j \in J \qquad (17)$$

$$y_t^i \in \{0,1\}, \quad \forall t \in T, i \in I. \qquad (18)$$

We propose another formulation that is based on the simple plant location formulation of DLSP, called IP4. In IP4 formulation, production variables are disaggregated not only in terms of periods in which produced items are consumed by demand, but also in terms of the CUs that they are produced by. $\Theta_{tt'}^{j}$ variables of IP3 are replaced by $\Phi_{tt'}^{ij}$, which gives the amount of product *j* produced using CU *i* in period *t* to be consumed in period t', and necessary changes are applied to constraints (20)–(22):

$$\mathbf{IP4}: \quad \text{minimize} \quad \sum_{t \in T} \sum_{i \in I} \left(f_t^i y_t^i + a_t^i x_t^i \right) - H \qquad (19)$$

$$\text{subject to} \quad \sum_{i \in I(j)} \sum_{t \le t'} \Phi_{tt'}^{ij} = d_{t'}^{j}, \qquad \forall t' \in T, j \in J \qquad (20)$$

$$\sum_{t' \ge t} \Phi_{tt'}^{ij} \le \alpha_i^j x_t^i, \qquad \forall i \in I, t \in T, j \in J(i) \qquad (21)$$

$$x_t^i \le \max_{j \in J(i)} \left\{ \frac{d_{tT}}{\alpha_i^j} \right\} y_t^i, \qquad \forall t \in T, i \in I \qquad (22)$$

$$x_t^i \ge 0, \qquad \forall t \in T, i \in I \qquad (23)$$

$$\Phi_{tt'}^{ij} \ge 0, \qquad \forall t, t' \in T, j \in J, i \in I \qquad (24)$$

$$y_t^i \in \{0,1\}, \qquad \forall t \in T, i \in I. \qquad (25)$$

Note that IP4 formulation has higher number of constraints and variables than IP3 formulation due to Equation (21), and the disaggregation of $\Theta_{tt'}^{j}$ into $\Phi_{tt'}^{ij}$, respectively.

3.2 Equivalence of Alternative Model Formulations

In order to show the equivalence of two linear mathematical models one can show any feasible solution of one model corresponds to some, also feasible, solution of the other model having the same objective value. This way one can conclude that feasible region of the first model is included in the feasible region of the second model. If the reverse also holds, then the models are said to be equivalent (Taşkın and Ekim, 2012). In this section, the equivalence will be shown explicitly between the linear relaxations of IP1 and IP2, IP2 and IP3, and IP3 and IP4. For convenience we name the linear relaxation of formulations as LP1, LP2, LP3 and LP4.

The only difference between IP1 and IP2's constraints is the structure of demand satisfaction constraints, i.e. Constraints (2) and (8) for IP1 and IP2, respectively.

Since initial inventories are zero, $s_0^j = 0$, and we represent the inventory variables of IP1 in terms of production variables and demand in IP2, the equivalance of LP1 and LP2 is straightforward.

Let us show the equivalence between LP2 and LP3. Consider constraints (9) and (15). For a given feasible solution (*x*, *y*) of any of the models, the other model is also feasible with respect to (9) and (15). Let $(\hat{x}_t^i, \hat{y}_t^i, \hat{\Theta}_{tt'}^j)$ be a feasible solution of LP3 formulation. Then, constraints (13) should hold for $\hat{\Theta}_{tt'}^j$. Constraints (26) are summed up versions of constraints (13) from 1 to *t*. We obtain Equation (27) when indices of two summations are swapped:

$$\sum_{z=1}^{t} \sum_{k=1}^{z} \hat{\Theta}_{kz}^j = \sum_{z=1}^{t} d_z^j, \qquad \forall t \in T,\ j \in J; \quad (26)$$

$$\sum_{k=1}^{t} \sum_{z=k}^{t} \hat{\Theta}_{kz}^j = d_{1t}^j, \qquad \forall t \in T,\ j \in J. \quad (27)$$

Constraints (14) should also hold for any feasible solution. We obtain Equation (28) when Constraints (14) are summed up from 1 to *t*. The combination of Equations (27) and (28) gives Equation (29). Using Equation (29) we can conclude that $\hat{x}$ satisfies constraints (8); and hence, $(\hat{x}_t^i, \hat{y}_t^i)$ is feasible with respect to LP2.

$$\sum_{k=1}^{t} \sum_{z=k}^{T} \hat{\Theta}_{kz}^j \le \sum_{k=1}^{t} \sum_{i \in I(j)} \alpha_i^j \hat{x}_k^i, \qquad \forall t \in T,\ j \in J; \quad (28)$$

$$d_{1t}^j = \sum_{k=1}^{t} \sum_{z=k}^{t} \hat{\Theta}_{kz}^j \le \sum_{k=1}^{t} \sum_{z=k}^{T} \hat{\Theta}_{kz}^j \le \sum_{k=1}^{t} \sum_{i \in I(j)} \alpha_i^j \hat{x}_k^i, \qquad \forall t \in T,\ j \in J. \quad (29)$$

The objective values of LP2 and LP3 are the same since both formulations share the same objective function.

Now, let $(\hat{x}_t^i, \hat{y}_t^i)$ be a solution of LP2 formulation. Unfortunately, reverse mapping of x_t^i variables of LP2 formulation into $\Theta_{tt'}^j$ variables of LP3 formulation is not unique. This is due to the fact that some production is done not to satisfy demand but occurs mandatorily due to co-production nature of the production environment. Since $\Theta_{tt'}^j$ variables only represent consumed production and there may be excess production, it is possible to shift production-consumption assignment in terms of $\Theta_{tt'}^j$ variables around. Therefore, using a simple first-in-first-out (FIFO) rule, it is possible to map any $(\hat{x}_t^i, \hat{y}_t^i)$ solution of LP2 formulation to a $(\hat{x}_t^i, \hat{y}_t^i, \hat{\Theta}_{tt'}^j)$ solution of LP3 formulation. The proposed algorithm is shown in Figure 3.

Equivalence between LP3 and LP4 follows from the relationship between $\Theta_{tt'}^j$ and $\Phi_{tt'}^{ij}$ variables. $\Phi_{tt'}^{ij}$ variables are CU disaggregated version of $\Theta_{tt'}^j$ variables. Let $(\hat{x}_t^i, \hat{y}_t^i, \hat{\Phi}_{tt'}^{ij})$ be a solution to LP4 formulation. Then by setting $\hat{\Theta}_{tt'}^j = \sum_{i \in I} \hat{\Phi}_{tt'}^{ij}$, $(\hat{x}_t^i, \hat{y}_t^i, \hat{\Theta}_{tt'}^j)$ will be a solution to LP3 formulation. Let $(\overline{x}_t^i, \overline{y}_t^i, \overline{\Theta}_{tt'}^j)$ be a solution to LP3 formulation. We need to map $\overline{\Phi}_{tt'}^{ij}$ arbitrarily from $\overline{\Theta}_{tt'}^j$ variables, and this mapping is not unique. This mapping can be done with an algorithm similar to the one given in Figure 3.

We have shown that the feasible regions of LP1 and LP2 formulations, LP2 and LP3 formulations, and LP3 and LP4 formulations are equal. Therefore, the feasible regions of all proposed models' linear relaxations are equal. We note that this result is in contrast to the DLSP, where the linear programming relaxation of simple plant location formulation is tighter than the basic formulation.

4 Algorithmic Improvements

Our preliminary computational tests reveal that none of the IP formulations can solve medium-size instances to optimality within a reasonable amount of time.

Therefore, we propose some improvements over the formulations in order to decrease the solution times. In this section, we discuss a separation algorithm that we propose for IP1, and then introduce similar algorithms for alternative formulations. We then provide a constructive heuristic that we propose to find an initial feasible solution to our problem. Finally, we propose an increasingly tighter set of lower bounds for the problem.

4.1 Valid Inequalities and Separation Algorithms

Valid inequalities, in general, improve the solution time required to solve integer programming formulations by narrowing the solution space. Although valid inequalities are not necessary to define the problem, they are satisfied by any feasible solution. Therefore, they could be violated by some fractional solutions of a branch and bound (B&B) tree but they never eliminate any integer feasible solution. However, in some cases there exists exponential number of valid inequalities with respect to the problem size. This makes it inefficient to include all valid inequalities in the formulation. Hence, it is computationally more efficient to add valid inequalities that are violated by the fractional solution of the node relaxation during the B&B search in order to improve the lower bound.

We propose valid inequalities, which are inspired by (*l*, *S*) inequalities given by Pochet and Wolsey (2006), in Proposition 2.

Proposition 2. Let X^{LP1} represent the feasible region of LP1. Also, let $l \in T$, $L = \{1,\ldots,l\}$, $S \subseteq L$,and $j \in J$ then the (*l*, *S*, *j*) inequality

$$\sum_{i \in I(j)} \sum_{q \in S} x_q^i \alpha_i^j \leq \sum_{q \in S} d_{ql}^j \left(\sum_{i \in I(j)} y_q^i \right) + s_l^j \qquad (30)$$

is valid for X^{IP1}.

Proof. Consider a point $(s, y) \in X^{LP1}$. If $\sum_{q \in S} \sum_{i \in I(j)} y_q^i = 0$, then as $\sum_{i \in I(j)} \sum_{q \in S} x_q^i = 0, s_l^j \geq 0$, the equality is satisfied. Otherwise let $t = min\{q \in S : \sum_{i \in I(j)} y_q^i > 0\}$. Then consider the following:

$$\sum_{i \in I(j)} \sum_{q \in S} x_q^i \alpha_i^j \leq \sum_{i \in I(j)} \sum_{q=t}^{l} x_q^i \alpha_i^j \leq d_{tl}^j + s_l^j \leq \sum_{q \in S} d_{ql}^j \left(\sum_{i \in I(j)} y_q^i \right) + s_l^j \quad (31)$$

First part of inequality (31) follows from non-negativity of $x_q^i \alpha_i^j$ terms, the definition of subset *S* and the time index *t*. The second part follows from flow conservation equations. Finally, the last part holds using $\sum_{i \in I(j)} y_t^i \geq 1$ and the non-negativity of y_t^i.

Remark. Inequalities of the form (30), do not give complete description of convex hull of IP1.

Moreover, the inventory variables in Equation (30) can be replaced by Equation (32), where $l \in T, L = \{1, \ldots, l\}, S \subseteq L$.

$$\sum_{i \in I(j)} \sum_{q \in L} x_q^i \alpha_i^j = d_{1l}^j + s_l^j \quad (32)$$

By using Equation (32) and Inequality (30), we obtain Inequality (33).

$$\sum_{i \in I(j)} \sum_{q \in L \setminus S} x_q^i \alpha_i^j + \sum_{q \in S} d_{ql}^j \left(\sum_{i \in I(j)} y_q^i \right) \geq d_{1l}^j \quad (33)$$

Note that valid inequalities of the form (33) are exponentially many. However, they can be separated by inspection using the algorithm given in Figure 4. A straightforward application of the algorithm leads to O(n^2) complexity whereas O($nlog(n)$) is doable by adapting the improvement proposed in Pochet and Wolsey (2006). Assume a fractional solution (x_q^{i*}, y_q^{i*}) to apply separation algorithm given in Figure 4. Note that this separation is exact, i.e., the algorithm finds all violated valid inequalities for a given solution.

Valid inequalities described in this section can be applied to IP3 and IP4 formulations using a similar logic as follows:

$$\sum_{q \in L \setminus S} \sum_{t' \geq q} \Theta_{qt'}^{j} + \sum_{q \in S} d_{ql}^{j} \left(\sum_{i \in I(j)} y_q^i \right) \geq d_{1l}^{j}, \quad (34)$$

$$\sum_{q \in L \setminus S} \sum_{t' \geq q} \sum_{i \in I(j)} \Phi_{qt'}^{ij} + \sum_{q \in S} d_{ql}^{j} \left(\sum_{i \in I(j)} y_q^i \right) \geq d_{1l}^{j}. \quad (35)$$

4.2 Pattern Fitting Heuristics

In this section we propose a heuristic to find a feasible solution of the problem, which we pass as an initial solution to the solver. Since the products can be produced by using different CUs, for a given CU, some of the demand of products that are produced within this CU may already be covered by the production of other CUs in previous time periods. For a given period *t*, we define the product coverage of a CU as the number of products whose demand for the same period is not covered yet by some previous production. Our heuristic works as follows: starting from the first period, the algorithm tries to cover all demand. The CU having the lowest cost to product coverage ratio is selected, and that CU is produced at an amount that covers all demand of products that CU is producing. Those products are marked as covered, and the algorithm selects the next CU with minimum cost to product coverage ratio until all products are covered for first period. Then, the excess production is reduced from demand for the next period and the algorithm continues for period 2, and so on to the last period. This pseudocode of our algorithm is given in Figure 5.

4.3 Lower Bound Calculation

In this section we derive a series of increasingly tighter lower bounds based on relaxations of our problem, and discuss how these can be incorporated within our Branch & Cut algorithm.

4.3.1 Dynamic Lot Sizing Relaxation

We first observe that the demand of each product must be satisfied, and we focus on calculating the cost of satisfying each product's demand independent of other products. Let us consider a DLSP relaxation of the problem for product *j*. Since demand (d_t^j) and holding cost (h_t^j) are given per product in our problem, these parameters can be used directly in the DLSP relaxation of product *j*. However, in our problem the fixed and variable costs of production (f_t^i and c_t^i, respectively) are specified per CU. Therefore, we need to calculate valid lower bounds for fixed and variable costs of product *j*.

Let us denote the fixed cost of product *j* in our DLSP relaxation by $f_t^{(j)}$. We define:

$$f_t^{(j)} = \min_{i \in I(j)} f_t^i. \qquad (36)$$

Similarly, let $c_t^{(j)}$ denote the variable cost of production of product *j* in our DLSP relaxation:

$$c_t^{(j)} = \min_{i \in I(j)} \frac{c_t^i}{\alpha_i^j} \qquad (37)$$

After this transformation, a lower bound on the cost of satisfying demand of product *j* can be calculated by solving DLSP for product *j* by using any DP algorithm for DLSP. Let $\mathrm{LB}_j^{(1)}$ denote the lower bound obtained for product *j*. Then, a valid lower bound for our problem, $\mathrm{LB}^{(1)}$, can be calculated as:

$$\mathrm{LB}^{(1)} = \max_{j \in J} \mathrm{LB}_j^{(1)} \qquad (38)$$

4.3.2 Co-Production Lot Sizing (CPLS) Relaxation

We note that the calculation of $\mathrm{LB}_j^{(1)}$ can be interpreted as defining a dummy CU for each product *j* whose fixed and variable costs are the minimum fixed and variable costs of the underlying CUs that can produce *j*. Since we calculate $f_t^{(j)}$ and $c_t^{(j)}$ independently, the cost structure of the dummy CU may not correspond to an existing CU, hence underestimating the total cost. In this section, we consider an extension of DLSP in which a single product can be produced via a number of CUs having different fixed and variable costs (CPLS). Formally, we define CPLS as follows:

CPLS. Given a set of CUs *I*, each with fixed cost, f_t^i, and variable cost, c_t^i; a single product *j*, with demand, d_t^j, and holding cost, h_t^j; production ratios, α_i^j; find a feasible production plan (such that all demands are satisfied on time) that has minimum total cost.

We observe that CPLS can be solved by a dynamic programming approach. In particular, let $C_{l,t}^j$ be the total cost of satisfying the demand for product *j* from period *l* to *t* from the production occured in period *l*.

$$C_{l,t}^j = \min_{i \in I(j)} \{ f_l^i + c_l^i d_{lt}^j \} + \sum_{k=l}^{t-1} h_k^j d_{k+1,t}^j \qquad (39)$$

Furthermore, let F_t^j be the minimum total cost when considering periods from 1 to *t*. Then, the following recursive function gives the current total minimum cost for product *j*.

$$F_t^j = \min_{k=1,\ldots,t} \{ F_{k-1}^j + C_{k,t}^j \} \qquad (40)$$

We can find a lower bound for the problem by calculating $\mathrm{LB}_j^{(2)}$ for each product by solving the recursive equation (40) in a DP-based algorithm and calculating the maximum among all products.

$$\mathrm{LB}^{(2)} = \max_{j \in J} \mathrm{LB}_j^{(2)} \quad (41)$$

4.3.3 Incorporating CPLS Lower Bounds in Branch & Cut

We observe that $\mathrm{LB}^{(1)}$ and $\mathrm{LB}^{(2)}$ can be used to calculate initial lower bounds on the problem. In this section, we discuss how we can incorporate an extension of $\mathrm{LB}^{(2)}$ within our Branch & Cut (B&C) algorithm. Let us add a binary variable w_i for each $i \in I$ into our model, which take on value 1 if and only if CU i is ever used for production in any period. We add constraints (42)–(43) to IP1.

$$y_i^t \le w_i \qquad \forall t \in T, i \in I \qquad (42)$$

$$w_i \le \sum_{t \in T} y_i^t \qquad \forall i \in I. \quad (43)$$

At any node of B&C tree, we check the current upper bounds of w_i variables. If the upper bound of w_i has been set to zero due to branching, then CU i cannot be used in the current B&C node or any of its children. Therefore, we can omit this CU while calculating $\mathrm{LB}^{(2)}$, thus increasing the lower bound. Furthermore, removal of a CU from the problem may lead to a disconnected set of products that are produced by disjoint CUs. We use this observation to further tighten the lower bound.

In particular, we construct a bipartite graph at each B&C node. The set of CUs whose w-variables have nonzero upper bound constitutes one part of the graph and products constitute the other part. We add an edge (i, j) if $\alpha_i^j > 0$. We then find connected components of this bipartite graph. We then calculate $\mathrm{LB}^{(2)}$ for each connected component, and add the lower bound obtained by each connected component to calculate a lower bound for the current B&C node. We add a local user cut that enforces the objective function value to be greater than or equal to the calculated lower bound. This local cut may increase the linear programming relaxation value at the current B&C node, and may be used by the solver to generate further cuts.

We note that the lower bound calculated by this approach gets tighter as more w_i variables' upper bounds are set to zero during branching. Therefore, we use branching priorities to force the solver to branch on w_i variables to obtain better lower bounds earlier in the B&C algorithm.

5 Computational Analysis

In this section we first give the data generation process for the test instances that we use. Then, we compare the efficacy of different MILP models using a commercial solver. Finally, we test effectiveness of the algorithmic improvements that we propose in Section 4.

We use ten different data sets that have different parameter settings as shown in Table 4. For each data set given in Table 4, we generate 10 random instances, resulting in 100 test instances in total. We set the time limit to 60 minutes per instance for all tests. We implement the models and algorithms in C++ programming language and use IBM CPLEX 12.9 in 64 bit mode for MILP solutions. Benchmark tests are performed on an Intel Core i7-3820 3.6 GHz machine with 32 GB RAM and 10 MB cache, running on Windows 10 operating system.

5.1 Data Generation Process

There is no available data library that we can directly use for the lot sizing problems in deliberated co-production literature. Therefore, we generate random data sets for experimentation. We use the data generation process used by Graves (1982) to generate a production planning problem where applicable. The procedure described in Graves (1982) has 20 products across three product families as shown in Table 2. Each product has a lower ($\underline{d}_t^j$) and an upper ($\overline{d}_t^j$) bound on demand for each period to be determined using uniform distribution. When more than 20 products are present in an instance, mod operation is used to determine the family of a product. For example, product 21 belongs to the first family whereas product 26 belongs to the second product family.

We use holding costs (h_t^j) and bounds on fixed costs of products ($\underline{fc}^j$ and $\overline{fc}^j$) as given by Graves (1982). The variable costs do not directly match with the ones given in Graves (1982), so we take variable costs of products, vc^j as 10, 15, and 20 for product families 1, 2, and 3, respectively. All these settings are provided in Table 3.

We create co-production units randomly while ensuring that each product is a part of at least one CU to ensure feasibility. A parameter named *density* (represented by δ) is used to determine the number of products inside each CU. Each CU can produce at least one and at most δ many products.

The fixed cost of a CU, f_t^i, is determined by taking %75 of the sum of the fixed costs of products that CU includes, i.e., $f_t^i = 0.75 \sum_{j \in J(i)} fc^j$, where fc^j is randomly generated in the interval $[\underline{fc}^j, \overline{fc}^j]$. Similarly, the variable cost of a CU, c_t^i, is taken as %90 of the sum of the variable costs of its products, i.e., $c_t^i = 0.9 \sum_{j \in J(i)} vc^j$. We create data sets with different sizes in terms of the time period, the number of products, the number of CUs and the density (δ) of CUs. The properties of these problem sets are summarized in Table 4.

5.2 Comparison of Alternative Model Formulations

We first implement MILP formulations directly without our proposed improvements. We solve all instances using CPLEX that uses 4 threads and has an MIPGap of 0. We give our computational results in Table 5. For each problem set and formulation we provide the number of test instances, out of 10, that give the best solution across different formulations and the average optimality gap at the end of time limit under the columns named '# of Best Sol.' and '% Gap' columns, respectively. For example, for PS 6, IP1 gives best solution for 9 and IP3 gives best solution for 1 of the 10 test instances. For some problem sets, the summation of best solutions across different formulations exceeds the total number of test instances as multiple formulations give equal solutions, which are also best solution, as in PS 1 and PS 2. We provide the total number of best solutions (*T*) and the average optimality gap (*A*) over all problem sets in the last row of Table 5.

As the number of time periods, the number of products and CUs increase, it becomes harder to solve the instances. As a result, the average gap increases as the problem sizes increase. Moreover, as the density of CUs increase, the average gap also increases except from PS 1 to PS 2 for which the average gap decreases for all formulations. Since the problem size is small in these two sets, we conclude that density does not significantly affect solution performance for small instances.

In terms of number of best solutions, IP1 gives the highest number of instances for five of the problem sets, IP2 gives the highest number of best solutions for three sets, IP4 gives the highest number for one set and for the remaining one all formulations are equal in terms of number of best solutions. When we analyze the average gap across different formulations, IP1 gives the best average gap for 8 out of 10 problem sets. Only for PS2, IP3 gives the best gap, and for PS1 both IP1 and IP3 give the best gap.

IP4 formulation performs the worst in terms of average gap values for medium and large problem sets (PS4 to PS10). Notice that IP4 formulation has the highest number of constraints for any given problem instance due to the constraints of the form (21). As density parameter increases from 3 to 4, only the number of constraints for IP4 increases due to constraint (21), which in turn significantly increases the problem size for IP4 formulation resulting in the worst performance.

We can conclude that IP1 dominates the other formulations in terms of both the average gap and the number of best solutions. Therefore, we continue our computational experiments by using IP1 as the MILP formulation.

5.3 Effectiveness of Proposed Algorithmic Improvements

In this section, we test the effect of adding valid inequalities by using the separation algorithm given in Section 4.1 to IP1 formulation. These experiments are based on the LP relaxation of IP1 formulation. We repeatedly solve LP relaxation of a model and add violated valid inequalities as needed. The algorithm for separating valid inequalities from a given fractional solution is given in Figure 4. Implementing this algorithm directly caused some numerical problems. For example, if a valid inequality that is violated by only a small amount is added to the formulation, CPLEX may not register it as a violated constraint due to numerical tolerances of CPLEX. This results in an infinite loop in some problem instances. Therefore, we restrict ourself to add valid inequalities that are violated by a specified value, called *epsilon*, to register as violated valid inequalities. Upon preliminary experimentation it is observed that the value of epsilon does not matter as long as it is not close to zero. Hence the value of epsilon is set to 20.

Another difficulty that we experienced while adding valid inequalities is that there is no standard mechanism to stop generating valid inequalities when these inequalities no longer improve current lower bound. In order to remedy this problem, we stop searching for valid inequalities when the percentage increase in the lower bound as a result of adding valid inequalities is less than 0.2%. We set the root algorithm parameter of CPLEX to dual simplex for this experiment.

We provide a summary of results in Table 6. Columns of Table 6 from left to right gives the average increase in the lower bound compared to LP relaxation (% Inc. in LB), the average gap between the LP relaxation solution and the best found feasible solution (% Gap–BS), and the average number of added valid inequalities (# of VI). As we see in Table 6, the valid inequalities and the separation algorithm proposed in Section 4.1 significantly improve the linear programming relaxation. In this experiment we do not see the effect of the size of the problem in terms of average increase in LB. On the other hand, as δ

parameter increases, which makes the CUs denser, the average increase in LB becomes higher. The average percentage gap of the LP relaxation solution with valid inequalities and the best known solution is promising as it might be as low as 4.95% with an average of 6.70%.

We also test IP1 formulation using three different settings and compare the results with the original MILP implementation. In the first setting we implement IP1 formulation without any improvements. In the second setting, separation algorithm is implemented for IP1 using callback structure of CPLEX. In the third setting, we apply the pattern fitting heuristic to generate an initial solution and implement separation algorithm. Finally, in the last setting we also implement CPLS procedure to find lower bounds at the fractional nodes. In these set of problems, we first turn off CPLEX cuts in order to see the net effect of our separation algorithm similar to the approaches given in Küçükyavuz and Pochet (2009) and Atamtürk and Küçükyavuz (2005). We provide the results in Table 7. Then, we turn on CPLEX cuts and run all experiments using CPLEX's default settings and provide the results in Table 8. For this experiment we set the *epsilon* value to 300 and 400 for δ values equal to 3 and 4, respectively.

In Tables 7 and 8, column blocks 'IP', 'Branch & Cut', 'Branch & Cut + PF', and ' Branch & Cut + PF + CPLS' give the results of MILP implementation, separation algorithm implementation, heuristics solution to be used as an initial solution, and finally all algorithms and heuristic implementations discussed in Section 4, respectively. For each problem set, '% Gap', 'Nnodes', '# Best Sol.' and 'Ncuts' columns provide the average optimality gap, the average number of B&B nodes considered, the number of best solutions found per problem set and the average number of valid inequalities generated, respectively. At the end of each column, we provide some aggregate results. For % Gap, Nnodes and Ncuts, we provide the average results over ten instances in each problem sets, and we give the total number for the # Best Sol. column.

We observe that when the CPLEX cuts are off, none of the problem instances can be solved to optimality within the time limit by all different settings. The third setting, where we use pattern fitting heuristic to provide an initial solution and then implement Branch & Cut algorithm (Branch & Cut + PF), gives the best average percentage gap calculated over all instances. However, when we check the optimality gaps for individual problem sets, we observe that in five of the ten problem sets, namely in PS4, PS6, PS7, PS8 and PS9, last setting, where we also use CPLS relaxation in fractional nodes to find a lower bound (Branch & Cut + PF + CPLS), gives the best gap.

In terms of number of best solutions found, direct implementation of the integer program (IP) is the best one with a total number of best solutions of 47 out of 100 instances. When we analyze the details for large instances, we observe that for PS9 and PS10 last setting dominates the other settings, while for PS8 and PS7 IP is better than the others. Although IP seems to give higher number of best solutions, the optimality gaps that it provides are very high compared to other settings. The reason for this high percentage is while IP's upper bounds are better than the others, the lower bounds found by IP are very loose.

When we analyze the results where CPLEX cuts are on, as it is expected, all solutions are improved accross all settings with IP being the most positively affected one. However, on the contrary of the previous table, IP provides the least number of best solutions for this setting. Moreover, PS2's all instances are solved to optimality within the time limit by all settings while 8 instances out of 10 are solved to optimality for PS1. We add a column for Table 8 called "# opt/Time" to provide the number of instances solved to optimality and the average CPU time the algorithms take to find an optimal solution for those instances. For PS1, Branch & Cut algorithm provides the minimum CPU time for the instances solved to optimality and it is followed by IP. For PS2, where all instances are solved to optimality, the average CPU time is the smallest for the third setting in which we use pattern fitting heuristic to provide an initial solution that we provide as an initial solution for the Branch & Cut algorithm. This setting is followed by the Branch & Cut algorithm without the heuristic, and IP yields the largest CPU time. For all other problem sets, none of the instances are solved to optimality within the time limit.

For the case where CPLEX cuts are on, last setting appears to be the best in terms of both the average optimality gap, 3.37% over all instances, and the total number of best solutions, 64 out of 100. When we analyze the problem sets indivdually in terms of gaps and number of best solutions, we observe that the last setting is better than the other settings for the sets with high density parameter.

6 Conclusion

In this paper we introduce a class of lot-sizing problem that appears in co-production systems. We define the problem in a controlled and deliberated co-production system, and show that the lot sizing problem in this type of system is NP-Hard in the strong sense. We develop four mixed-integer programming formulations. Then, we show that the LP relaxation of all four formulations are equal. We propose a separation algorithm based on a set of valid inequalities as a solution method, a heuristic that aims to provide an initial solution to be given to a solver and another heuristic that is used to obtain fast upper bounds for fractioanl solutions.

There is no benchmark data set available for lot sizing problems in a co-production environment; therefore, we generate ten different data sets to be used for computational analysis. We test the efficacy of the proposed MILP formulations, the separation algorithm and the heuristics on this data set. Based on our computational experiments, we conclude that, among all four formulations, the MILP formulation that includes the inventory variables explicitly is the best one in terms of average gap provided at the end of a time limit. Moreover, the setting where the separation algorithm, the initial solution heuristic and the CPLS relaxation to calculate lower bounds are used improves the solution quality in terms of the average optimality gap and the number of best solutions found over all test instances.

Future avenues for research include using polyhedral analysis to find the convex hull of the problem, strengthening the mixed-integer program by developing some valid inequalities that will tighten the solution space and developing heuristics based on dynamic programming approaches applicable to regular lot-sizing algorithms.

Acknowledgements

This research is supported by TUBITAK Project No: 116M555.

Short Bio

Bahadır Pamuk is a Ph.D. candidate at the Department of Industrial Engineering, Boğaziçi University, İstanbul, Turkey. He received his B.Sc. and M.Sc. degrees in Industrial Engineering from Boğaziçi University in 2016 and 2018 respectively. He is a member of Flexible Automation and Intelligent Manufacturing Systems (BUFAIM) Laboratory since 2014. His research interests include combinatorial optimization, large-scale optimization, and flexible automation.

Semra Ağralı is a professor of industrial engineering at Faculty of Engineering, MEF University, İstanbul, Turkey. Her current research interests include energy systems optimization, large-scale optimization problems in supply chain management and scheduling. Her publications have appeared in *European Journal of Operational Research, OMEGA, IEEE Transactions on Power Systems, Journal of Operational Research Society*, and other journals. She is the recipient of *Goodeve Medal 2020* given by the Journal of the OR Society. Her research has been supported by The Scientific and Technological Research Council of Turkey.

Z. Caner Taşkın is a professor of operations research at Boğaziçi University, İstanbul, Turkey. His research focuses on decomposition algorithms for large-scale mixed integer programming arising in application areas such as supply chain planning, telecommunications, graph theory and medicine. His research has appeared in journals including *Operations Research, INFORMS Journal on Computing, IISE Transactions, European Journal of Operational Research.* He is also a recipient of Institute of Industrial and Systems Engineers (IISE) Pritsker Doctoral Dissertation Award, Turkish Science Academy’s Young Scientist Award (BAGEP) and Best Application Paper in the 2020 *IISE Transactions* Focus Issue on Operations Engineering and Analytics.

Banu Kabakulak is an Assistant Professor of Industrial Engineering at İstanbul Bilgi University, İstanbul, Turkey since 2019. She received her double major B.Sc. degrees in Industrial Engineering and Mathematics from Boğaziçi University, İstanbul, Turkey, in 2007, and M.Sc. and Ph.D. degrees in Industrial Engineering from Boğaziçi University, in 2010 and 2018, respectively. Her research interests are telecommunications, mathematical programming, large-scale optimization, and decomposition methods. She is the recipient of *Operations Engineering & Analytics Best Application Paper Award 2020* given by the IISE Transactions Journal.

References

Ağralı, S. (2012). A dynamic uncapacitated lot-sizing problem with co-production. *Optimization Letters* *6* (6), 1051–1061.

Atamtürk, A. and S. Küçükyavuz (2005). Lot sizing with inventory bounds and fixed costs: Polyhedral study and computation. *Operations Research* *53* (4), 711–730.

Bitran, G. R. and S. Dasu (1992). Ordering policies in an environment of stochastic yields and substitutable demands. *Operations Research* *40* (5), 999–1017.

Bitran, G. R. and S. M. Gilbert (1994). Co-production processes with random yields in the semiconductor industry. *Operations Research* *42* (3), 476–491.

Brahimi, N., N. Absi, S. Dauzère-Pérès, and A. Nordli (2017). Single-item dynamic lot-sizing problems: An updated survey. *European Journal of Operational Research* *263* (3), 838–863.

Brahimi, N., S. Dauzere-Peres, N. M. Najid, and A. Nordli (2006). Single item lot sizing problems. *European Journal of Operational Research* *168* (1), 1–16.

Euromak (2020). All molds.

Garey, M. R. and D. S. Johnson (1979). Computers and intractability: a guide to the theory of NP-completeness. 1979. *San Francisco, LA: Freeman 58*.

Graves, S. C. (1982). Using lagrangean techniques to solve hierarchical production planning problems. *Management Science 28* (3), 260–275.

Krarup, J. and O. Bilde (1977). Plant location, set covering and economic lot size: An O(mn)-algorithm for structured problems. In *Numerische Methoden bei Optimierungsaufgaben Band 3*, pp. 155–180. Springer.

Küçükyavuz, S. and Y. Pochet (2009). Uncapacitated lot sizing with backlogging: the convex hull. *Mathematical Programming 118* (1), 151–175.

Öner, S. and T. Bilgiç (2008). Economic lot scheduling with uncontrolled co-production. *European Journal of Operational Research 188* (3), 793–810.

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

Rafiei, R., M. Nourelfath, J. Gaudreault, L. A. De Santa-Eulalia, and M. Bouchard (2015). Dynamic safety stock in co-production demand-driven wood remanufacturing mills: A case study. *International Journal of Production Economics 165*, 90–99.

Robinson, P., A. Narayanan, and F. Sahin (2009). Coordinated deterministic dynamic demand lot-sizing problem: A review of models and algorithms. *Omega 37* (1), 3–15.

Stern, T. (2006). Seminar in theoretical computer science. Set Cover Problem.

Suzanne, E., N. Absi, and V. Borodin (2020). Towards circular economy in production planning: Challenges and opportunities. *European Journal of Operational Research*.

Taşkın, Z. C. and T. Ekim (2012). Integer programming formulations for the minimum weighted maximal matching problem. *Optimization Letters 6* (6), 1161–1171.

Taşkın, Z. C. and A. T. Ünal (2009). Tactical level planning in float glass manufacturing with co-production, random yields and substitutable products. *European Journal of Operational Research 199* (1), 252–261.

Vidal-Carreras, P. I., J. P. Garcia-Sabater, and J. R. Coronado-Hernandez (2012). Economic lot scheduling with deliberated and controlled coproduction. *European Journal of Operational Research 219* (2), 396–404.

Wagelmans, A., S. Van Hoesel, and A. Kolen. Economic lot sizing: an O (n log n) algorithm that runs in linear time in the Wagner-Whitin case. *Operations Research 40*.

Wagner, H. M. and T. M. Whitin (1958). Dynamic version of the economic lot size model. *Management science 5* (1), 89–96.

Yuan, Z., J. Bi, and Y. Moriguichi (2006). The circular economy: A new development strategy in China. *Journal of Industrial Ecology 10* (1-2), 4–8.

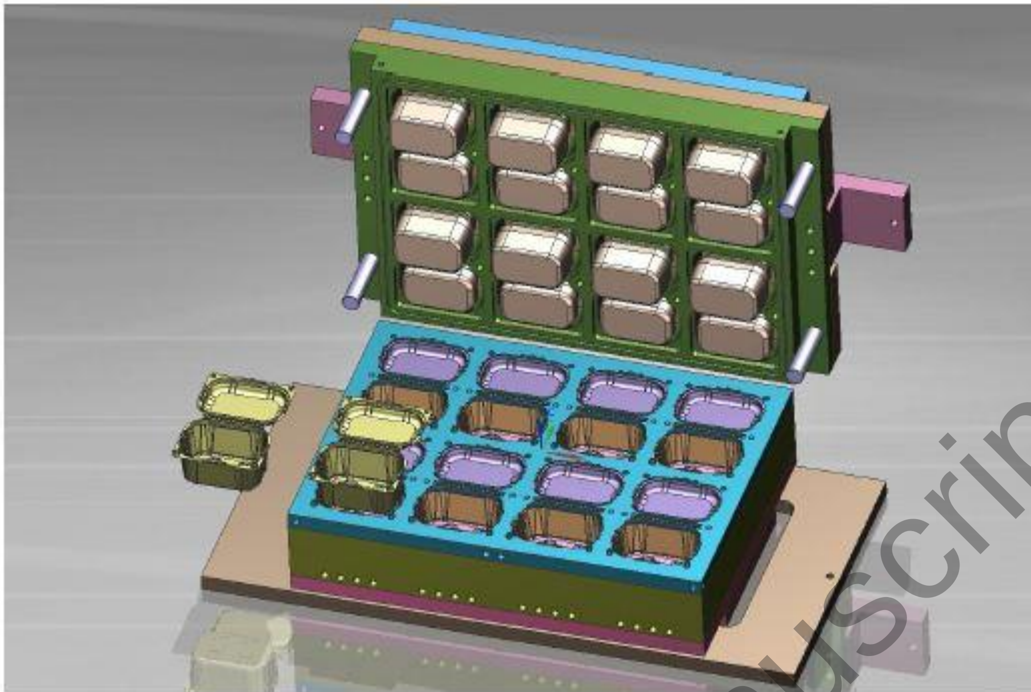

Fig. 1 Co-production in plastic mould industry with different fixtures (Euromak, 2020)

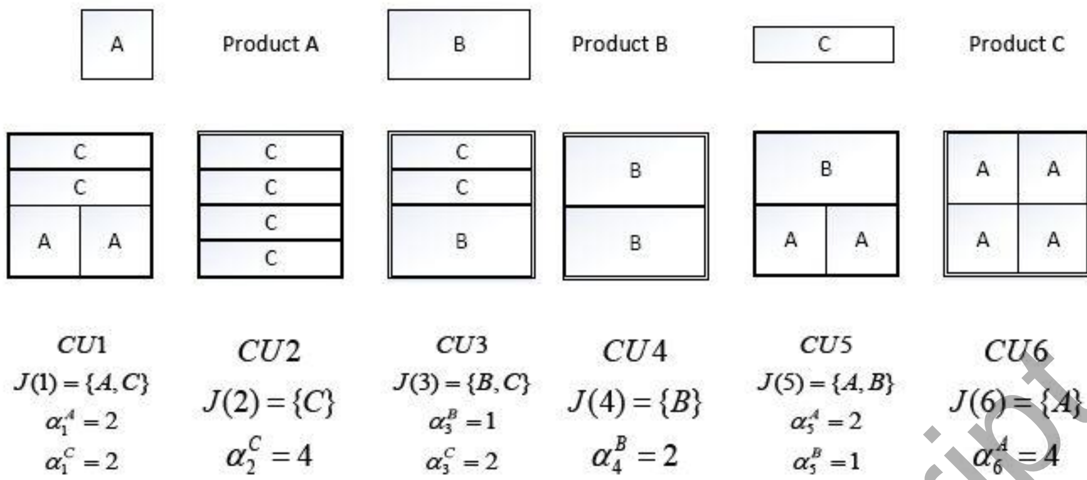

Fig. 2 An Example for Deliberated and Controlled Co-production

```
for Each product $j \in J$ do
  Make copy of demand vector $d_t^j$ into $D_t$ for all $t \in T$
  Make copy of production vector $\sum_{i \in I(j)} x_t^i \alpha_i^j$ into $P_t$ for all $t \in T$
  for Each period $t \in T$ do
    for $p \in \{1, ..., T\}$ do
      if $P_p > D_t$ then
        $\Theta_{pt}^j = D_t$
        $P_p = P_p - D_t$
        break
      else
        $\Theta_{pt}^j = P_p$
        $D_t = D_t - P_p$
      end if
    end for
  end for
end for
```

Fig. 3 Algorithm for mapping $\hat{\Theta}_{jtt'}$ from $\hat{x}_t^i$ using FIFO.

for Each product $j \in J$ **do**
  for $l = 1, ..., T$ **do**
    Calculate $D_l^j = \sum_{q=1}^{l} min\left\{ \sum_{i \in I(j)} x_q^{i*} \alpha_i^j, d_{ql}^j \left(\sum_{i \in I(j)} y_q^{i*} \right) \right\}$
    if $D_l^j < d_{1l}^j$ **then**
      return j, $L = \{1, ..., l\}$, $S = \left\{ q \in L : \sum_{i \in I(j)} x_q^{i*} \alpha_i^j > d_{ql}^j \left(\sum_{i \in I(j)} y_q^{i*} \right) \right\}$
    end if
  end for
end for

Fig. 4 Algorithm for (*l*, *S*, *j*) Separation.

Initialize R_j, C_j, X_{it}, Y_{it}, $S_{jt} = 0$, $minratio = \infty$
for Period $t \in T$ **do**
  for Product $j \in J$ **do**
    $R_j = R_j + d_t^j$
  end for
  if $R_j \leq 0$ **then**
    $C_j = 1$
  else
    $C_j = 0$
  end if
  while $C_j = 0 \quad \exists j \in J$ **do**
    for CU $i \in I$ **do**
      for Product $j \in J(i)$ **do**
        $ratio = R_j / \alpha_i^j$
        if $ratio > X_{it}$ **then**
          $X_{it} = ratio$
        end if
      end for
      for Product $j \in J(i)$ **do**
        $S_{jt} = X_{it} * \alpha_i^j - R_j$
      end for
      Define $J'(i) = \{j \quad | \quad R_j > 0, j \in J(i)\}$
      if $J'(i) \neq \emptyset$ **then**
        $CTCR = \{f_i^t + X_{it} * p_i^t + \sum_{j \in J(i)} h_j^t * S_{jt}\}/|J'(i)|$
      end if
      if $CTCR < minratio$ **then**
        $minratio = CTCR$
        $CUindex = i$
      end if
    end for
    if $CUindex \geq 0$ **then**
      $Y_{it} = 1$
      for Product $j \in J(i)$ **do**
        $C_j = 1$
        $R_j = R_j - X_{CUindex,t} * \alpha_{CUindex}^j$
      end for
    end if
  end while
end for
return X_{it}, Y_{it}, S_{jt}

Fig. 5 Pattern Fitting Heuristic.

Table 1 Notation and Definitions.

T	**: Set of time periods, indexed by t**
J	: Set of products, indexed by j
I	: Set of co-production units (CUs), indexed by i
$J(i)$	: Set of products produced by CU i
$I(j)$	: Set of CUs that produce product j
α_i^j	: Co-produced amount of product j, when one unit of CU i is produced
f_t^i	: Fixed cost of CU i at period t
c_t^i	: Variable cost of CU i at period t
h_t^j	: Holding cost of product j at period t
h_{tk}^j	: Cumulative holding cost of product j between periods t and k
d_t^j	: Demand of product j at period t
d_{tk}^j	: Cumulative demand of product j between periods t and k

Table 2 Product Families and Demand Data Used in Experimentation.

Family	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3
Product (*j*)	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
$\underline{d}_t^j$	20	40	15	25	80	80	15	40	40	55	20	20	30	30	20	30	30	50	60	40
$\overline{d}_t^j$	40	60	25	65	120	120	25	60	60	85	40	40	50	50	40	70	50	100	90	80

Table 3 Holding Cost, Fixed Cost, and Variable Cost Used in Experimentation.

Family	h^j	$\underline{fc}^j$	$\overline{fc}^j$	vc^j
1	1	50	150	10
2	1.75	100	200	15
3	1.5	100	200	20

Table 4 Problem Set Descriptions.

Problem Set	$\lvert T \rvert$	$\lvert J \rvert$	$\lvert I \rvert$	δ
PS 1	12	10	20	3
PS 2	12	10	20	4
PS 3	24	40	120	3
PS 4	24	40	120	4
PS 5	24	40	200	3
PS 6	24	40	200	4
PS 7	36	40	120	3
PS 8	36	40	120	4
PS 9	36	40	200	3
PS 10	36	40	200	4

Table 5 Summary of Results for Alternative Model Formulations

	# of Best Sol.				Gap			
PS	**IP1**	**IP2**	**IP3**	**IP4**	**IP1**	**IP2**	**IP3**	**IP4**
PS1	8	8	8	9	0.17	0.51	0.17	0.29
PS2	10	10	10	10	0.03	0.19	0.00	0.12
PS3	0	6	2	2	4.72	5.71	5.00	5.08
PS4	1	6	1	2	4.73	6.61	5.31	5.74
PS5	7	2	1	0	4.88	7.48	5.36	12.47
PS6	9	0	1	0	4.52	8.52	5.28	16.64
PS7	4	6	0	0	4.67	7.72	6.04	12.38
PS8	5	4	1	0	4.78	8.55	5.74	14.17
PS9	10	0	0	0	5.55	12.41	6.92	26.08
PS10	10	0	0	0	5.64	15.01	9.40	41.27
	64	42	24	23	3.97	7.27	4.92	13.42

Table 6 Summary of Results for Separation Algorithm for IP1 Formulation

	% Inc. in LB	% Gap-BS	# of VI
PS 1	15.83	4.95	690
PS 2	17.79	5.43	634
PS 3	15.14	6.68	11573
PS 4	19.71	7.64	11820
PS 5	14.46	6.49	11704
PS 6	20.58	6.68	11992
PS 7	15.06	6.87	24688
PS 8	19.32	7.41	25607
PS 9	15.39	7.18	25737
PS 10	20.84	7.63	26615
Avg.	17.41	6.70	15106

Table 7 Summary of Results for IP1 in Different Settings where CPLEX cuts are off.

	IP				Branch & Cut			
PS	Gap (%)	Nnodes	# Best Sol.	Ncuts	Gap (%)	Nnodes	# Best Sol.	Ncuts
PS1	10.31	11862973	3	-	1.49	1776741	7	5806
PS2	5.87	24238609	7	-	0.22	336158	10	4223
PS3	18.13	2138837	9	-	7.59	2936	0	35544
PS4	20.95	2178332	8	-	8.15	2614	0	39565
PS5	17.54	1992753	6	-	6.66	1419	0	27285
PS6	21.73	2044701	1	-	6.76	921	0	30297
PS7	18.18	1725201	6	-	6.74	530	1	49844
PS8	20.85	2067092	7	-	7.14	391	0	61926
PS9	20.14	1908900	0	-	7.04	7	2	41405
PS10	23.70	1631485	0	-	7.17	41	0	52042
	A: 17.74	*A: 5178888*	*T: 47*	*A: -*	*A: 5.89*	*A: 212176*	*T: 20*	*A: 34794*
	Branch & Cut + PF				**Branch & Cut + PF + CPLS**			
PS	**Gap (%)**	**Nnodes**	**# Best Sol.**	**Ncuts**	**Gap (%)**	**Nnodes**	**# Best Sol.**	**Ncuts**
PS1	1.45	1885518	8	5927	4.80	1932068	6	4955
PS2	0.24	325875	9	4276	3.53	600523	9	4014
PS3	7.29	3095	1	36645	7.77	4462	0	29256
PS4	8.26	2450	0	39606	7.63	3858	2	33161
PS5	6.22	1364	3	27312	6.25	1140	1	23379
PS6	6.14	989	3	31577	5.69	1252	6	29849
PS7	6.72	475	2	48268	6.70	128	1	44320

	IP				Branch & Cut			
PS8	7.20	394	1	61109	6.64	199	2	58262
PS9	6.72	83	2	42130	6.62	0	6	40801
PS10	6.49	26	5	52112	6.72	21	5	51703
	A: 5.67	*A: 222027*	*T: 34*	*A: 34896*	*A: 6.23*	*A: 254365*	*T: 38*	*A: 31970*

Table 8 Summary of Results for IP1 in Different Settings.

	IP					Branch & Cut				
PS	**Gap (%)**	**Nnodes**	**# Best Sol.**	**# opt/ Time**	**Ncuts**	**Gap (%)**	**Nnodes**	**# Best Sol.**	**# opt/ Time**	**Ncuts**
PS1	0.10	633444	9	8/473.3	-	0.11	625484	10	8/442.7	36.4
PS2	0.00	115255	10	10/260.9	-	0.00	90614	10	10/199.8	9.8
PS3	4.52	10816	0	-	-	4.22	10704	1	-	2034.9
PS4	4.41	21653	0	-	-	4.13	20959	2	-	1293.2
PS5	4.68	1765	0	-	-	4.55	1850	0	-	1488.4
PS6	4.35	3871	0	-	-	4.30	3612	0	-	1132.1
PS7	4.52	2636	3	-	-	4.57	2524	1	-	4636.4
PS8	4.59	4742	0	-	-	4.47	4587	1	-	4010.1
PS9	5.19	11	2	-	-	5.33	0	2	-	3084.3
PS10	5.37	94	0	-	-	5.13	135	0	-	2765.8
	A: 3.77	*A: 79429*	*T: 24*	*A: -*	*A: -*	*A: 3.68*	*A: 76047*	*T: 27*	*A: -*	*A: 2049*
	Branch & Cut + PF					**Branch & Cut + PF + CPLS**				
PS	**Gap (%)**	**Nnodes**	**# Best Sol.**	**# opt/ Time**	**Ncuts**	**Gap (%)**	**Nnodes**	**# Best Sol.**	**# opt/ Time**	**Ncuts**
PS 1	0.12	650897	9	8/518.9	34.2	0.67	899873	8	8/1065.6	34.0
PS 2	0.00	93763	10	10/185.6	9.8	0.00	137233	10	10/220.1	10.1
PS 3	4.18	10848	1	-	1866.2	3.61	82762	8	-	2110.5
PS 4	4.32	18472	1	-	1333.0	3.72	138274	7	-	1195.8
PS 5	4.07	1605	8	-	1482.8	4.32	8182	2	-	1201.2

	IP					Branch & Cut				
PS 6	3.63	3420	1	-	1138.7	3.32	15538	9	-	1078.9
PS 7	4.37	2883	3	-	4278.8	4.53	8765	3	-	6432.7
PS 8	4.54	5751	0	-	4379.4	3.83	18390	9	-	11652.3
PS 9	5.04	0	5	-	3169.5	5.30	0	1	-	2178.7
PS 10	4.46	136	3	-	2779.5	4.45	8	7	-	2067.1
	A: 3.47	*A: 947032*	*T: 41*	*A: -*	*A: 2047*	*A: 3.37*	*A: 944579*	*T: 64*	*A: -*	*A: 2796*