Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 716248, 9 pages
http://dx.doi.org/10.1155/2015/716248

Research Article

Supply Chain Batching Problem with Identical Orders and Lifespan

Shanlin Li,[1] Maoqin Li,[1] and Hong Yan[2]

[1]*Department of Mathematics, Taizhou University, Taizhou, Zhejiang 317000, China*
[2]*Department of Logistics and Maritime Studies, Hong Kong Polytechnic University, Kowloom, Hong Kong*

Correspondence should be addressed to Shanlin Li; sxtyxdljz@163.com

Received 22 December 2014; Revised 20 April 2015; Accepted 21 April 2015

Academic Editor: Yuri Vladimirovich Mikhlin

In the real world, there are a large number of supply chains that involve the short lifespan products. In this paper, we consider an integrated production and distribution batch scheduling problem on a single machine for the orders with a short lifespan, because it may be cheaper or faster to process and distribute orders in a batch than to process and distribute them individually. Assume that the orders have the identical processing time and come from the same location, and the batch setup time is a constant. The problem is to choose the number of batches and batch sizes to minimize the total delivery time without violating the order lifespan. We first give a backward dynamic programming algorithm, but it is not an actually polynomial-time algorithm. Then we propose a constant time partial dynamic programming algorithm by doing further research into the recursion formula in the algorithm. Further, using the difference characteristics of the optimal value function, a specific calculating formula to solve the problem with the setup time being integer times of the processing time is obtained.

1. Introduction

In the past two decades, research in the area of supply chain management (SCM) has increased significantly. An important aspect of SCM concerns integrated planning of production and distribution processes since the increasing globalization and strong competition force companies not only to compete with others with regard to prices or quality, but with regard to reliability and timeliness of the deliveries as well. Thus, a coordinated planning of production and transportation is vital for the success of a business; especially if products with short lifespans are involved, the coordination of related operations becomes a challenging issue, such as food (Tarantilis and Kiranoudis [1]; Arbib et al. [2]; Chen et al. [3]; Farahani et al. [4]; Méndez et al. [5]; Amorim et al. [6]; Bilgen and Günther [7]; Kaplan and Rabadi [8]; Shirvani et al. [9]), yoghurt products (Kopanos et al. [10]; Bilgen and Çelebi [11]), industrial chemicals (Geismar et al. [12]; Armstrong et al. [13]; Viergutz and Knust [14]), ready-mix concrete (García et al. [15]; Garcia and Lozano [16, 17]), medical specimens (Zangeneh-Khamooshi et al. [18]), or also daily newspapers (van Buer et al. [19]). The integrated production and distribution scheduling problems can be classified into two types based on whether there is a lifespan as the specific parameter of the problem. Since the focus of this paper is on problems with lifespan constraints, we do not review papers that study problems without lifespan constraints, which can be found in Chen [20].

Chen et al. [3] consider production scheduling and vehicle routing with time windows for perishable food products to maximize the expected total profit of the supplier. The demands at retailers are assumed stochastic and perishable goods will deteriorate once they were produced. They propose a nonlinear mathematical model, which is NP-hard, and give a solution algorithm composed of the constrained Nelder-Mead method and a heuristic for the vehicle routing with time windows to solve the complex problem. Farahani et al. [4] study a similar problem but with multiple production lines and sequence dependent setup times and costs. The production scheduling problem is solved through an mixed-integer linear programming modeling approach which is based on a block planning formulation.

Méndez et al. [5] and Amorim et al. [6] study the integrated batch scheduling and vehicle routing problem and the integrated lot sizing and scheduling and vehicle routing problem for perishable goods respectively. They propose mixed-integer programming models and analyse computational results of the models, respectively.

Bilgen and Günther [7] present an integrated production scheduling and truck routing model for a fruit juice supply chain. They considered different transportation modes and described the production system based on the block planning approach which establishes cyclical production patterns with regard to the definition of setup families.

Kaplan and Rabadi [8] focus on the scheduling of perishable products on parallel machines. Each job has a due date, which is the preferred delivery date of the retailers and might be violated subject to a penalization as lateness penalty, and there is a strict deadline imposed by the retailer that should not be exceeded. The objective is to minimize the total penalty cost. They propose a mixed-integer programming model and analyse computational results of the model. Shirvani et al. [9] study a similar problem but with job dependent holding cost. They propose also a mixed integer programming model and a heuristic solution beside an iterated greedy algorithm is developed to generate good and feasible solutions for the problem.

Kopanos et al. [10] consider production scheduling and distribution planning in the yoghurt processing industry with the limited shelf life of intermediate mixes in the aging stage. The model proposed by them decides the assignment of transportation trucks to processing sites-distribution center in every period as well as transportation load for every truck. They impose material balance and logistics operations constraints. Three different transportation modes and min/max truck capacity are taken into consideration. Bilgen and Çelebi [11] study a similar problem but with maximizing the benefit by considering the shelf life dependent pricing component and costs such as processing, setup, storage, overtime, backlogging, and transportation costs. A mixed-integer linear programming model is developed for the considered problem. The efficiency and applicability of the proposed model and approach are demonstrated in a case study for a dairy manufacturing company in Turkey.

Motivated by applications of certain time-sensitive chemical compounds, Geismar et al. [12], Armstrong et al. [13], and Viergutz and Knust [14] consider problems where orders expire within a certain time frame once they are produced and hence must be delivered before they expire. In Geismar et al.'s problem, orders to be delivered in the same batch are produced as a lot together and once a lot is completed the shipment must depart immediately and deliver the orders to their destinations within a given time frame. They propose a metaheuristic for this strongly NP-hard problem and evaluate the performance of the heuristic computationally. In the Armstrong et al.'s problem, all the orders are processed and delivered in a single shipment in a prespecified sequence. In addition to a common expiration time frame, each order has an individual time window within which the order must be delivered. The objective is to choose a subset of the orders to be delivered such that the total demand of the chosen orders is maximized. The authors show that their problem is at least ordinarily NP-hard and give a branch-and-bound exact algorithm and a heuristic for the problem. Viergutz and Knust's problem is similar to Armstrong et al.'s. They refine the Armstrong et al.'s algorithm and extend the model for handling delays of the production start as well as for variable production and distribution sequences and give heuristic and evaluate their performance computationally.

Garcia and Lozano [16] consider a ready-mix concrete production and vehicle scheduling problem with identical parallel machines and multiple customers. They give a new approach for constructing the min-cost network flow problem. They also consider a dual problem with the objective of minimizing the number of vehicles used subject to the constraint that a given number of orders are covered. They give an exponential-time exact solution approach and a partial branch and bound heuristic. More specifically, Garcia and Lozano [17] study a similar problem but with an ideal due date for each order. The revenue of a delivered order decreases with the deviation from the ideal due date. This problem is strongly NP-hard. They give a tabu search based heuristic.

Zangeneh-Khamooshi et al. [18] consider a medical specimens collection problem with a central depot and multiple customers which is modeled as a multishift vehicle routing problem with windows and cycle times, which is a strongly NP-hard problem. They give an new version of the vehicle routing problem with time windows that minimizes the total cycle time of the orders. When the courier's schedule is allowed to vary, they propose an algorithm to determine feasible routes and schedules of the available couriers and evaluate their performance computationally.

van Buer et al. [19] consider a newspaper printing and delivery problem with a single machine and multiple customers. There are sequence-dependent setup times between orders in the production part and each order has a delivery deadline that must be met. There is a fixed cost for using a vehicle and a vehicle if used can be used to cover multiple trips. The objective is to minimize the total fixed and variable transportation cost. The authors give several heuristics and evaluate their performance computationally.

In this paper, we study a supply chain involving the product with a short lifespan which consists of a plant and a set of customer groups, where the production facility of the plant consists of a single machine and the set of customer groups comes from the same location (e.g., a distribution hub). At the single machine a single product with a limited lifespan is produced. The product lifespan specifies that the product expires after the end of its production. The orders of the product from all of customers are identical. Our problem is to schedule and deliver these identical orders in batches, since it may be cheaper or faster to process and deliver orders in a batch than to process and deliver them individually. A setup time is required at the start of the schedule and on each occasion when the machine switches from processing orders in one batch to orders in another batch (e.g., the time of changing a tool or to clean the machine). The delivery time of an order coincides with the delivery time of the last scheduled orders in its batch and all orders in this batch have the same delivery time. For a given number of orders, we want

to choose batch sizes so as to minimize the sum of the delivery times of the orders without violating the order lifespan.

Formally, there is a set of n orders from the same location with identical processing time p and a limited lifespan L, $J = \{j_1, \ldots, j_n\} = \{1, \ldots, n\}$, to be processed on a single machine. In a given schedule, for each job $j \in J$, we denote C_j to be its completion time and $D_j = C_j + T$ delivery time of order j, where T is the transportation time from the plant to order j's destination. The problem is, given a setup time S, to find the number of batches k and batch sizes b_i satisfying $\sum_{i=1}^{k} b_i = n$, so as to minimize $\sum_{j=1}^{n} D_j = \sum_{j=1}^{n}(C_j + T) = \sum_{i=1}^{k} b_i \sum_{j=1}^{i}(S + b_j p) + nT$ without violating the order lifespan, where all of p, L, T, and S are positive integers.

In a given batch schedule, to respect the lifespan constraint of the order, it is necessary that $(b_i - 1)p + T \le L$ for each batch sizes b_i, where $(b_i - 1)p + T$ is the time from the completion time point of the first scheduled order in ith batch to its delivery time point. Let B denote the maximal integer less than or equal to rational number $(L - T)/p + 1$. Since nT in the objective function $\sum_{j=1}^{n} D_j = \sum_{j=1}^{n} C_j + nT$ is a constant, our problem can be translated to the problem: find the number of batches k and batch sizes b_i satisfying $\sum_{i=1}^{k} b_i = n$ and $b_i \le B$ for $i = 1, \ldots, k$, so as to minimize $\sum_{j=1}^{n} C_j = \sum_{i=1}^{k} b_i \sum_{j=1}^{i}(S + b_j p)$. The problem is referred to as $1|p_j = p;\ B;\ S - \text{batch}|\sum C_j$.

This paper is organized as follows. In Section 2, we give a backward dynamic programming algorithm to solve problem $1|p_j = p;\ B;\ S - \text{batch}|\sum C_j$, which is not an actually polynomial-time algorithm. In Section 3, we first show the relation between optimal solution and the first order difference of the optimal objective function of the number of orders and investigate the properties of the first order difference. Then a partial dynamic programming algorithm to solve the problem in constant time is given. A specific calculating formula to solve the problem in case $S = vp$ is shown in Section 4. In Section 5, we present some numerical examples to show the effectiveness of our proposed algorithms. Section 6 contains a conclusion and a discussion of some possible extensions.

2. Dynamic Programming Algorithm

For ease of presentation, we sequence the orders according to nonincreasing indices. Any solution of problem $1|p_j = p;\ B;\ S - \text{batch}|\sum C_j$ is of the form

$$BS : Sn_k \cdots (n_{k-1} + 1)\, Sn_{k-1} \cdots (n_{k-2} + 1) \cdots Sn_1 \cdots 1, \quad (1)$$

where k is the number of batches, $1 \le n_1 < n_2 < \cdots < n_k = n$, and batch sizes $b_1 = n_k - n_{k-1}, \ldots, b_{k-1} = n_2 - n_1$, $b_k = n_1$ satisfied with $b_i \le B$ for $i = 1, \ldots, k$. Every solution BS corresponds to a objective function value

$$\begin{aligned} F(BS) &= \sum_{j=1}^{n} C_j = \sum_{i=1}^{k} b_i \sum_{j=1}^{i} \left(S + b_j p\right) \\ &= \sum_{i=1}^{k} n_i \left(S + \left(n_i - n_{i-1}\right) p\right), \end{aligned} \quad (2)$$

where $n_0 = 0$.

In order to solve the batch sizing problem, we obviously have to find a constant k and a sequence of indices $1 \le n_1 < n_2 < \cdots < n_k = n$ such that the above objective function value is minimized. Clearly, problem $1|p_j = p;\ B;\ S - \text{batch}|\sum C_j$ is a trivial matter when $S \le p$ or $B < 2$. We have the result as follows.

Theorem 1. *If $S \le p$ or $B < 2$, then for problem $1|p_j = p;\ B;\ S - batch|\sum C_j$ there exists an optimal solution in which $k = n$ and $n_i = i$ for $i = 1, \ldots, n$; that is, $b_1 = \cdots = b_k = 1$.*

By Theorem 1, we assume that $S > p$ and $B \ge 2$ hereafter. Below we give a backward dynamic programming algorithm to solve the problem $1|p_j = p;\ B;\ S - \text{batch}|\sum C_j$.

Algorithm 2 (dynamic programming algorithm). Let $F(j)$ denote the minimum the sum of the completion times for j-orders batching problem containing orders $1, \ldots, j$. The initialization is

$$F(0) = 0 \quad (3)$$

and the recursion for $j = 1, \ldots, n$ is

$$\begin{aligned} F(j) = \min \{ & j\left[S + \left(j - i\right)p\right] + F(i) \mid \max\{0, j - B\} \\ & \le i \le j - 1 \}. \end{aligned} \quad (4)$$

The minimization selects a batch $\{j, \ldots, i + 1\}$ which dose not violate the product lifespan to insert at the start of the previous schedule containing orders $i, \ldots, 1$. Batch $\{j, \ldots, i + 1\}$ completes at time $S + (j - i)p$, and the processing of the batches containing jobs $i, \ldots, 1$ is delayed by time $S + (j - i)p$ as a result of the insertion. The optimal solution value is then equal to $F(n)$. Under the most natural implementation, the algorithm requires $O(nB)$ time.

However, it is not an actually polynomial-time algorithm, since the usual definition of the input size is the sum of the logarithms of the input parameters n, S, p, and B. In the following sections, we will do further research into the recursion formula: $F(j) = \min\{j[S + (j - i)p] + F(i) \mid \max\{0, j - B\} \le i \le j - 1\} = j[S + (j - t)p] + F(t)$, so as to determine completely the optimal successor t for every j in an polynomial-time of the input size.

3. Partial Dynamic Programming Algorithm

In this section, we first show the relation between optimal solution and the first order difference of the optimal objective function in terms of the number of orders and investigate the properties of the first order difference. Then we give a partial dynamic programming algorithm to solve the problem in constant time. For each order j, if order t with $\max\{0, j - B\} \le t \le j - 1$ such that

$$F(j) = j\left[S + \left(j - t\right)p\right] + F(t) \quad (5)$$

holds, we call order t the optimal successor of order j and denote the optimal successor t of j by SUCC(j). For $0 \le i \le j - 1$, set

$$F(j, i) = j\left[S + \left(j - i\right)p\right] + F(i). \quad (6)$$

Thus, we have

$$F(j) = \min_{i=\max\{0,j-B\}}^{j-1} F(j,i). \quad (7)$$

Note that the fact that $i+1$ is better (worse) than i as a successor of j, that is, $F(j,i+1)-F(j,i) = j[S+(j-i-1)p]+F(i+1)-j[S+(j-i)p]+F(i) = F(i+1)-F(i)-jp < (>)0$, is equivalent to $F(i+1)-F(i) < (>)jp$, which indicates that the optimal successor of j is related to the first order difference sequence of the optimal objective function of the number of orders. We denote the first order difference sequence of the optimal objective function of the number of orders by $\Delta(i) = F(i+1)-F(i)$ for $i = 0, 1, \ldots, +\infty$. We define that $\Delta(-1) = 0$. Then, $i+1$ is better (worse) than i as a successor of j if and only if $\Delta(i) < (>)jp$. To break ties when choosing the successor, we assume that if $\Delta(i) = jp$, that is, $F(j,i+1) = F(j,i)$, then $i+1$ is better than i as a successor of j. The following lemma shows an important characteristic of the first order difference sequence.

Lemma 3. *If $S > p$, then all of the following hold:*

(i) $p \le \Delta(j) - \Delta(j-1) \le 2p$ *for* $j = 1, 2, \ldots$;

(ii) *if* $SUCC(j+1) = s_{j+1}$ *and* $SUCC(j) = s_j$, *then* $(j+1) - s_{j+1} \ge j - s_j$.

Proof. We prove this lemma by induction. Simple calculations yield the following:

$\mathrm{SUCC}(1) = 0$ and $\Delta(0) = F(1) - F(0) = S + p$,

$F(2) = \min\{F(2,1), F(2,0)\} = \min\{2(S+p)+S+p, 2(S+2p)\} = 2(S+2p)$, and $\mathrm{SUCC}(2) = 0$,

$\Delta(1) = F(2) - F(1) = S + 3p > \Delta(0) = S + p$ and $\Delta(1) - \Delta(0) = 2p$ and $2 - 0 > 1 - 0$.

Thus, (i) and (ii) hold for $j = 1$. Now suppose that they hold when $j \le h$ for some positive integer h, that is, $p \le \Delta(t) - \Delta(t-1) \le 2p$ and $(t+1) - s_{t+1} \ge t - s_t$ for $t = 1, 2, \ldots, h$. We need to show that they hold when $j = h+1$. Assume that $\mathrm{SUCC}(h+1) = t$ $(0 \le t \le h)$. By $\Delta(1) - \Delta(0) = 2p$ and $p \le \Delta(j) - \Delta_1(j-1)$ for $j = 2, \ldots, h$, we have that $(h+1)p \le \Delta(h)$. Equation $(h+1)p = \Delta(h)$ does not hold. Otherwise, inequalities $\Delta(0) < \cdots < \Delta(h-1) < (h+1)p$ imply that 1 is better than 0, 2 better than $1, \ldots, h$ better than $h-1$ in order as a successor of $h+1$. The result is that $\mathrm{SUCC}(h+1) = h$; that is, $F(h+1) = (h+1)(S+p) + F(h)$. Then $(h+1)p = \Delta(h) = F(h+1) - F(h) = (h+1)(S+p) + F(h) - F(h) = (h+1)(S+p)$. This is a contradiction. Thus, we have that $(h+1)p < \Delta(h)$. Noting the fact that $\{\Delta(j)\}_{j=-1}^{h}$ is a strictly monotone increasing sequence, there is $0 \le t_0 \le h$ such that $\Delta(t_0 - 1) \le (h+1)p < \Delta(t_0)$. There are two cases to consider: (a) $h+1-t < B$; (b) $h+1-t = B$.

Case (a) $(h+1-t < B)$. Since the inequalities $\Delta(-1) < \Delta(0) < \cdots < \Delta(t_0 - 1) \le (h+1)p$ imply that t_0 is better than $t_0 - 1$, $t_0 - 1$ better than $t_0 - 2, \ldots, 1$ better than 0 in order and the inequalities $(h+1)p < \Delta(t_0) < \cdots < \Delta(h)$ imply that t_0 is better than t_0+1, t_0+1 better than $t_0+2, \ldots, h-1$ better than h in order as a successor of $(h+1)$, along with $h+1-t < B$, we have that $t = t_0 = \mathrm{SUCC}(h+1)$. This implies that $\mathrm{SUCC}(h+1) = t$ and $h+1-t < B$ if and only if $\Delta(t-1) \le (h+1)p < \Delta(t)$.

If $t_0 = 0$, then $\mathrm{SUCC}(h) = 0$. If $t_0 = 1$, either $\mathrm{SUCC}(h) = 0$ or $\mathrm{SUCC}(h) = 1$ follows from the fact that $hp < (h+1)p < \Delta(t_0) < \cdots < \Delta(h)$. If $t_0 > 1$, due to $p \le \Delta(t_0 - 1) - \Delta(t_0 - 2)$, either $\mathrm{SUCC}(h) = t_0$ or $\mathrm{SUCC}(h) = t_0 - 1$ follows from the fact that $\Delta(t_0 - 2) \le hp < (h+1)p < \Delta(t_0)$. In sum, if $\mathrm{SUCC}(h+1) = t$, then either $\mathrm{SUCC}(h) = t$ or $\mathrm{SUCC}(h) = t-1$. Similarly, we have that if $\mathrm{SUCC}(h+1) = t$, then either $\mathrm{SUCC}(h+2) = t$ or $\mathrm{SUCC}(h) = t+1$. This implies that (ii) holds. To end the proof of (i), there are four cases to consider: (a1) $\mathrm{SUCC}(h) = \mathrm{SUCC}(h+1) = \mathrm{SUCC}(h+2) = t$, (a2) $\mathrm{SUCC}(h) = t-1$, $\mathrm{SUCC}(h+1) = \mathrm{SUCC}(h+2) = t$, (a3) $\mathrm{SUCC}(h) = \mathrm{SUCC}(h+1) = t$, $\mathrm{SUCC}(h+2) = t+1$, and (a4) $\mathrm{SUCC}(h) = t-1$, $\mathrm{SUCC}(h+1) = t$, $\mathrm{SUCC}(h+2) = t+1$.

Case (a1) $(\mathrm{SUCC}(h) = \mathrm{SUCC}(h+1) = \mathrm{SUCC}(h+2) = t)$. In this case,

$$\begin{aligned} F(h) &= h[S + (h-t)p] + F(t), \\ F(h+1) &= (h+1)[S + (h+1-t)p] + F(t), \\ F(h+2) &= (h+2)[S + (h+2-t)p] + F(t). \end{aligned} \quad (8)$$

Thus, we have that $\Delta(h) = F(h+1) - F(h) = S + (2h+1-t)p$, $\Delta(h+1) = F(h+2) - F(h+1) = S + (2h+3-t)p$, and

$$\Delta(h+1) - \Delta(h) = 2p. \quad (9)$$

Case (a2) $(\mathrm{SUCC}(h) = t-1, \mathrm{SUCC}(h+1) = \mathrm{SUCC}(h+2) = t)$. In this case,

$$\begin{aligned} F(h) &= h[S + (h-t+1)p] + F(t-1), \\ F(h+1) &= (h+1)[S + (h+1-t)p] + F(t), \\ F(h+2) &= (h+2)[S + (h+2-t)p] + F(t). \end{aligned} \quad (10)$$

Thus, we have that $\Delta(h) = F(h+1) - F(h) = S + (h+1-t)p + F(t) - F(t-1) = S + (h+1-t)p + \Delta(t-1)$, $\Delta(h+1) = F(h+2) - F(h+1) = S + (2h+3-t)p$, and

$$\Delta(h+1) - \Delta(h) = (h+2)p - \Delta(t-1). \quad (11)$$

Since $\mathrm{SUCC}(h+1) = \mathrm{SUCC}(h+2) = t$, the inequalities $\Delta(t) > (h+2)p > (h+1)p \ge \Delta(t-1)$ hold. This implies $\Delta(h+1) - \Delta(h) = (h+2)p - \Delta(t-1) \ge p$. By $t-1 \ge 0$, $1 \le t \le h$ hold. Due to the induction assumption, along with $\Delta(t) > (h+2)p$, we have that $\Delta(h+1) - \Delta(h) = (h+2)p - \Delta(t-1) \le \Delta(t) - \Delta(t-1) \le 2p$.

Case (a3) $(\mathrm{SUCC}(h) = \mathrm{SUCC}(h+1) = t,\ \mathrm{SUCC}(h+2) = t+1)$. In this case,

$$\begin{aligned} F(h) &= h[S + (h-t)p] + F(t), \\ F(h+1) &= (h+1)[S + (h+1-t)p] + F(t), \\ F(h+2) &= (h+2)[S + (h+2-t-1)p] + F(t+1). \end{aligned} \quad (12)$$

Thus, we have that $\Delta(h) = F(h+1) - F(h) = S + (2h+1-t)p$, $\Delta(h+1) = F(h+2) - F(h+1) = S + (h+1-t)p + F(t+1) - F(t) = S + (h+1-t)p + \Delta(t)$, and

$$\Delta(h+1) - \Delta(h) = \Delta(t) - hp. \tag{13}$$

Since $\mathrm{SUCC}(h) = \mathrm{SUCC}(h+1) = t$, the inequalities $\Delta(t) > (h+1)p > hp$ hold. This implies $\Delta(h+1) - \Delta(h) = \Delta(t) - hp \geq p$. Also, we have that $\Delta(h+1) - \Delta(h) = \Delta(t) - hp \leq 2p$. Otherwise, $\Delta(t) - hp > 2p$ implies $\Delta(t) > (h+2)p$. This contradicts with the fact that $\Delta(t) \leq (h+2)p < \Delta(t+1)$.

Case (a4) ($\mathrm{SUCC}(h) = t-1$, $\mathrm{SUCC}(h+1) = t$, $\mathrm{SUCC}(h+2) = t+1$). In this case,

$$\begin{aligned} F(h) &= h\left[S + (h+1-t)p\right] + F(t-1), \\ F(h+1) &= (h+1)\left[S + (h+1-t)p\right] + F(t), \\ F(h+2) &= (h+2)\left[S + (h+1-t)p\right] + F(t+1). \end{aligned} \tag{14}$$

Thus, we have that

$$\Delta(h+1) - \Delta(h) = \Delta(t) - \Delta(t-1). \tag{15}$$

Since $1 \leq t \leq h$, along with the induction assumption, the inequalities $p \leq \Delta(h+1) - \Delta(h) \leq 2p$ hold. This ends the proof for Case (a).

Case (b) ($h+1-t = B$). The inequalities $\Delta(t_0 - 1) \leq (h+1)p < \Delta(t_0)$ still hold, where $0 \leq t_0 \leq h$. Clearly, $t \geq t_0$. Otherwise, $t < t_0$. Since $h + 1 - t_0 < B$ which means that the product lifespan is not violated and $\Delta(t_0 - 1) \leq (h+1)p < \Delta(t_0)$, $\mathrm{SUCC}(h+1) = t_0$. This contradicts with $\mathrm{SUCC}(h+1) = t$. Assume that $\mathrm{SUCC}(h) = t'$. There are two cases to consider: (b1) $h - t' = B$ and (b2) $h - t' < B$.

Case (b1) ($h - t' = B$). By $\Delta(t'+1) - \Delta(t') \geq p$ and $hp < \Delta(t')$, we have that $(h+1)p < \Delta(t'+1)$. This, along with $((h+1) - t') > B$, implies that $\mathrm{SUCC}(h+1) = t' + 1$; that is $t' = t - 1$. similarly, $\mathrm{SUCC}(h+2) = t+1$. Similar to the proof of Case (a4), (i) and (ii) hold for the case.

Case (b2) ($h - t' < B$). Clearly, $t' \leq t_0$. By $(h+1) - t_0 \leq (h+1) - t' \leq B$, along with $\Delta(t_0 - 1) \leq (h+1)p < \Delta(t_0)$, we have that $t = t_0$. Meanwhile we claim $t' = t_0$. Otherwise $t' < t_0$. Then $(h+1) - t = (h+1) - t_0 \leq h - t' < B$. This contradicts with $h + 1 - t = B$. Similarly, by $h + 1 - t = B$, $\mathrm{SUCC}(h+2) = t + 1 = t_0 + 1$. Similar to the proof of Case (a3), (i) and (ii) hold for the case. This ends the proof of the lemma. □

The lemma, as well as four formulas yielded in the proof of the lemma, is very important for our batch sizing problem. $\Delta(0) = S + p$ implies that 0 is the optimal successor of $1, 2, \ldots, \lfloor (S+p)/p \rfloor$, where $\lfloor (S+p)/p \rfloor$ denotes the maximal integer less than rational number $(S+p)/p$ and $\lfloor (S+p)/p \rfloor \leq B$. $p \leq \Delta(j) - \Delta(j-1) \leq 2p$ implies that each positive integer must be the optimal successor of one or two positive integers. If $\mathrm{SUCC}(h) = t$ and $h - t = B$, then $\mathrm{SUCC}(j) = j - B$ for $j = h+1, h+2, \ldots$. By (ii) in Lemma 3, we may assume that h^* is the minimal integer such that $\mathrm{SUCC}(h^*) = t^*$ and $h^* - t^* = B$. Then $j < h^*$ and $\mathrm{SUCC}(j) = t$ imply that $j - t < B$ and then that $j < h^*$ and $\mathrm{SUCC}(j) = t$ hold if and only if $\Delta(t-1) \leq hp < \Delta(t)$. By (ii) in Lemma 3, if h^* may be found by recursion (4) in a constant time, then the our problem is solvable in a constant time, since $\mathrm{SUCC}(j) = j - B$ for $j > h^*$.

Now we define that $N_k = \{j \mid$ the optimal number of batches of j-orders batching problem is equal to $k\}$ for $k = 1, 2, \ldots$, called k-batches case set of orders and sequence integers in N_k in increasing natural order. Then $(N_1, \ldots, N_k, \ldots)$ is a partition of the set of position integers. Let $C_{1,i} = i$ for $i = 1, \ldots, |N_1|$. Then $N_1 = (C_{1,1}, \ldots, C_{1,|N_1|})$. We define that $C_{k,i} = \{j \mid \mathrm{SUCC}(j) \in C_{k-1,i}\}$ for $k = 2, 3, \ldots$ and $i = 1, \ldots, |N_1|$, called the ith periodic set of N_k, and sequence integers in $C_{k,i}$ in increasing natural order. Then $N_k = (C_{k,1}, \ldots, C_{k,|N_1|})$ and $(C_{1,1}, \ldots, C_{1,|N_1|}, \ldots, C_{k,1}, \ldots, C_{k,|N_1|}, \ldots)$ is a partition of the set of position integers. If we can find a upper bound of $|C_{k,i}|$, the upper bound of $j - t$ will be estimated, where $j \in C_{k,i}$, $t \in C_{k-1,i}$ and $\mathrm{SUCC}(j) = t$. The following lemma shows how to do this.

Lemma 4. *Let $S = vp + r$ and $q_{k-1} = \sum_{t=1}^{k-1} |N_t|$, where $q_0 = 0$, $0 \leq r < p$, and v are integers. Then for all of k satisfying with $j - t < B$ where $j \in N_k$ and $\mathrm{SUCC}(j) = t$, all of the following hold:*

(i) $|C_{ki}| = k$ *for* $i = 1, \ldots, v$,

(ii) $\Delta(q_{k-1} + (i-1)k + k) - \Delta(q_{k-1} + (i-1)k) = (k+1)p$ *for* $i = 1, \ldots, v$.

Proof. By $S = vp + r$ and $\Delta(0) = S + p = (v+1)p + r$, $N_1 = \{1, \ldots, v\}$ if $r = 0$; $N_1 = \{1, \ldots, v+1\}$ if $r > 0$. Let $C_{1,v+1} = \emptyset$ if $r = 0$ and $C_{1,v+1} = v + 1$ if $r > 0$. We prove this lemma by induction. Clearly, $C_{1i} = i$ for $i = 1, \ldots, v$. If $C_{1,v+1} = v + 1$, $\Delta(i) - \Delta(i-1) = 2p$ for $i = 1, \ldots, v$ follows from (9), since $\mathrm{SUCC}(0) = \mathrm{SUCC}(1) = \cdots = \mathrm{SUCC}(v+1)$. If $C_{1,v+1} = \emptyset$, $\Delta(i) - \Delta(i-1) = 2p$ for $i = 1, \ldots, v-1$ follows from (9), since $\mathrm{SUCC}(0) = \mathrm{SUCC}(1) = \cdots = \mathrm{SUCC}(v)$. By $\mathrm{SUCC}(v-1) = \mathrm{SUCC}(v) = 0$, $\mathrm{SUCC}(v+1) = 1$, and (13), $\Delta(v) - \Delta(v-1) = \Delta(0) + (v-1)p = 2p$. Thus (i) and (ii) hold when $k = 1$. Now suppose that they hold when $k = h$ for some positive integer h. We need to show that they hold when $k = h+1$. For any $i \in \{1, \ldots, v\}$, by result (ii) with $k = h$: $\Delta(q_{h-1} + (i-1)h + h) - \Delta(q_{h-1} + (i-1)h) = (h+1)p$ and $j \in C_{(h+1)i}$ if and only if $\Delta(q_{h-1} + (i-1)h) \leq jp < \Delta(q_{h-1} + (i-1)h + h)$, we have that $|C_{(h+1)i}| = h + 1$. Thus the result (i) holds when $k = h + 1$.

Now we rewrite $C_{h+1,i}$ as $(b+1, \ldots, b + (h+1))$, where $b = q_h + (i-1)(h+1)$. By $|C_{h,i}| = h$ and $|C_{h+1,i}| = h + 1$, every integer in $C_{h,i}$ must be the optimal successor of some integer in $C_{h+1,i}$ and there is a integer, say $a + t_0$ where $a = q_{h-1} + (i-1)h$, in $C_{h,i}$ such that it is the optimal successor of two integers in $C_{h+1,i}$. So, we have that

$$\begin{aligned} &\mathrm{SUCC}(b+t) = a + t \quad \text{for } t = 1, \ldots, (t_0 - 1), \\ &\mathrm{SUCC}(b + t_0) = \mathrm{SUCC}(b + t_0 + 1) = a + t_0, \\ &\mathrm{SUCC}(b+t) = a + t - 1 \\ &\qquad \text{for } t = (t_0 + 2), \ldots, (h+1). \end{aligned} \tag{16}$$

By (15),

$$\Delta(b+t)-\Delta(b+t-1)=\Delta(a+t)-\Delta(a+t-1),\quad \text{for } t=1,\ldots,(t_0-1). \tag{17}$$

By (11),

$$\Delta(b+t_0)-\Delta(b+t_0-1)=(b+t_0+1)p-\Delta(a+t_0-1). \tag{18}$$

By (13),

$$\Delta(b+t_0+1)-\Delta(b+t_0)=\Delta(a+t_0)-(b+t_0)p. \tag{19}$$

By (15),

$$\Delta(b+t)-\Delta(b+t-1)=\Delta(a+t-1)-\Delta(a+t-2),\quad \text{for } t=(t_0+2),\ldots,(h+1). \tag{20}$$

Equations (17), (18), (19), and (20), along with the induction assumption, imply $\Delta(b+h+1)-\Delta(b)=\Delta(a+h)-\Delta(a)+p=(h+2)p$; that is, the result (ii) holds when $k=h+1$. This ends the proof of the lemma. □

In the following we give a partial dynamic programming algorithm to solve the problem $1|p_j=p;\ B;\ S-\text{batch}|\sum C_j$.

Algorithm 5 (partial dynamic programming algorithm). Let $F(j)$ denote the minimum the sum of the completion times for j-orders batching problem containing orders $1,\ldots,j$. The initialization is

$$F(0)=0, \tag{21}$$

and for $j=1,\ldots,h^*$ the recursion is

$$F(j)=\min\{j[S+(j-i)p]+F(i)\mid 0\le i\le j-1\}; \tag{22}$$

for $j>h^*$ the recursion is

$$F(j)=j[S+Bp]+F(j-B), \tag{23}$$

where h^* is the minimal integer such that $\text{SUCC}(h^*)=t^*$ and $h^*-t^*=B$.

Theorem 6. *Algorithm 5 finds an optimal solution of problem* $1|p_j=p;\ B;\ S-batch|\sum C_j$ *in constant time.*

Proof. By (ii) in Lemma 3, $j-s_j$ is monotonic increasing, where $\text{SUCC}(j)=s_j$. By (i) in Lemma 4, $|C_{ki}|=k$ for $k\ge 1$ and $i=1,\ldots,v$, where $S=vp+r$. Noting the fact that $\text{SUCC}(j_{k+1,v})=j_{k,v}$, where $j_{k+1,v}$ and $j_{k,v}$ are the last orders in $C_{k+1,v}$ and C_{ki}, respectively, we have that $(j_{k+2,v}-j_{k+1,v})-(j_{k+1,v}-j_{kv})\ge v$ for $k\ge 1$. This, along with $v\ge 1$, implies that the recursion: $F(j)=\min\{j[S+(j-i)p]+F(i)\mid 0\le i\le j-1\}$ for $j=1,\ldots,h^*$ finishes in $O(B)$ time. Thus, Algorithm 5 finds an optimal solution of the problem in constant time. □

In the following section, we will show a specific calculating formula for the solution of the problem with $S=vp$.

4. Specific Calculating Formula in Case $S=vp$

Assume $S=vp$. By (i) in Lemma 4, we may obtain a partition of the set of j-orders with $j\le h^*$ as follows: $N_1=\{1,\ldots,v\}$, $N_2=\{v+1,\ldots,3v\},\ldots$, and $N_k=\{\sum_{t=1}^{k-1}tv+1,\ldots,\sum_{t=1}^{k}tv\},\ldots$. For example, if $v=3$, $C_{11}=\{1\}$, $C_{12}=\{2\}$, $C_{13}=\{3\}$, and $N_1=\{1,2,3\}$, $C_{21}=\{4,5\}$, $C_{22}=\{6,7\}$, $C_{33}=\{8,9\}$, and $N_2=\{4,\ldots,9\};\ldots$.

For ease of presentation, we denote by (k,i,w) the wth order in the ith periodic set C_{ki} of the kth batch case set N_k, where all k, i, and w with $k\ge 1$, $1\le i\le v$, and $1\le w\le k$ are integers; that is,

$$(k,i,w)=\sum_{t=1}^{k-1}tv+(i-1)k+w=\frac{(k-1)k}{2}v+(i-1)k+w, \tag{24}$$

where $\sum_{t=1}^{0}tv=0$. Equation (24) establishes a one-to-one correspondence between $\{(k,i,w)\mid k\ge 1,\ 1\le i\le v,\ 1\le w\le k\}$ and the set of positive integers. For example, if $v=3$, $(4,3,2)=\sum_{t=1}^{4-1}tv+(3-1)4+2=28$ and $101=33v+2=((7\times 8)/2)v+(3-1)8+1=(8,3,1)$. We define $(0,0,0)=0$; $(k,i,w)=(k,i-1,k)$ if $w=0$ and $i>1$; $(k,i,w)=(k-1,v,k-1)$ if $w=0$ and $i=1$. The following lemma describes the special properties of first order difference in case $S=vp$.

Lemma 7. *Assume $S=vp$. Then all of the following hold.*

(i) *For each order (k,i,w) with $(k,i,w)\le h^*$, $\Delta(k,i,w)-\Delta(k,i,w-1)=p$ if $w<k$ and $\Delta(k,i,w)-\Delta(k,i,w-1)=2p$ if $w=k$, where $k\ge 1$, $1\le i\le v$, and $1\le w\le k$.*

(ii) *Let $B=mv+r$, where m and $0\le r<v$ are nonnegative integers. Then $h^*=(m+1,r,m+1)$.*

Proof. We prove result (i) by induction. By $\text{SUCC}(i)=0$ for $i=0,1,\ldots,v$, along with (9), we have that $\Delta(1,i,1)-\Delta(1,i,0)=\Delta(i)-\Delta(i-1)=2p$ for $i=1,\ldots,v-1$. By $\text{SUCC}(v-1)=\text{SUCC}(v)=0$ and $\text{SUCC}(v+1)=1$, along with (13), we have $\Delta(1,v,1)-\Delta(1,v,0)=\Delta(v)-\Delta(v-1)=\Delta(0)-(v-1)p=(v+1)p-(v-1)p=2p$. Thus, result (i) holds for $k=1$. Now suppose that it holds when $k\le h$; that is, $\Delta(h,i,w)-\Delta(h,i,w-1)=p$ if $w<h$ and $\Delta(h,i,w)-\Delta(h,i,w-1)=2p$ if $w=h$ for $i=1,\ldots,v$. We need to show it for $k=h+1$.

For each i with $1\le i\le v$, by the induction assumption, $\text{SUCC}(h+1,i,w)=(h,i,w)$ for $w=1,\ldots,h$ and $\text{SUCC}(h+1,i,h+1)=(h,i,h)$ for $w=h+1$ follow from the fact that $\Delta(h,i,w)-\Delta(h,i,w-1)=p$ if $w<h$ and $\Delta(h,i,w)-\Delta(h,i,w-1)=2p$ if $w=h$ for $i=1,\ldots,v$. Since $(h+1,i,0)=(h+1,i-1,h+1)$ if $i>1$ and $(h+1,i,0)=(h,v,h)$, along with the induction assumption, we have that $\text{SUCC}(h+1,i,0)=(h,i-1,h)$ if $i>1$ and $\text{SUCC}(h+1,i,0)=(h-1,v,h-1)$. This implies that $\text{SUCC}(h+1,i,w)$, $\text{SUCC}(h+1,i,w+1)$, and $\text{SUCC}(h+1,i,w+2)$ are different from each other for $0\le w\le h-2$. By (15) and the induction assumption, we have that

$\Delta(h+1,i,w)-\Delta(h+1,i,w-1)=\Delta(h,i,w)-\Delta(h,i,w-1)=p$ for $w=1,\ldots,h-1$. Since $\mathrm{SUCC}(h+1,i,h-1)=(h,i,h-1)$ and $\mathrm{SUCC}(h+1,i,h)=\mathrm{SUCC}(h+1,i,h+1)=(h,i,h)$, $\Delta(h+1,i,h)-\Delta(h+1,i,h-1)=(h+1,i,h+1)p-\Delta(h,i,h-1)=[\sum_{t=1}^{h}tv+(i-1)(h+1)+(h+1)]p-\Delta(h,i,h-1)$ follows from (11). By $\Delta(0,0,0)=(v+1)p$, (ii) in Lemma 4 and the induction assumption, we have that $\Delta(h,i,h-1)=[(v+1)p]+[v(2p)+v(3p)+\cdots+v(hp)]+[(i-1)(h+1)p+(h-1)p]=[\sum_{t=1}^{h}tv]p+[(i-1)(h+1)]p+hp$. This implies that $\Delta(h+1,i,h)-\Delta(h+1,i,h-1)=p$. Note the fact that $(h+1,i,h+1)+1=(h+1,i+1,1)$ if $i<v$ and $(h+1,i,h+1)+1=(h+2,1,1)$ if $i=v$. We have that $\mathrm{SUCC}(h+1,i,h)=\mathrm{SUCC}(h+1,i,h+1)=(h,i,h)$ and $\mathrm{SUCC}[(h+1,i,h+1)+1]=(h,i,h)+1$. By (19), $\Delta(h+1,i,h+1)-\Delta(h+1,i,h)=\Delta(h,i,h)-(h+1,i,h)p=\Delta(h,i,h)-[\sum_{t=1}^{h}tv+(i-1)(h+1)+h]p$. By $\Delta(0,0,0)=(v+1)p$, (ii) in Lemma 4 and the induction assumption, we have that $\Delta(h,i,h)=[(v+1)p]+[v(2p)+v(3p)+\cdots+v(hp)]+[i(h+1)p]=[\sum_{t=1}^{h}tv]p+[(i-1)(h+1)]p+(h+2)p$. This implies that $\Delta(h+1,i,h+1)-\Delta(h+1,i,h)=2p$. This ends the proof of result (i).

By (ii) in Lemma 3 and result (i), order h^* satisfying with $\mathrm{SUCC}(h^*)=t^*$ and $h^*-t^*=B$ must be one of the set of $\{(k,i,k)\mid k\geq 1,\ 1\leq i\leq v\}$. Since $\mathrm{SUCC}(m+1,r,m+1)=(m,r,m)$ and $(m+1,r,m+1)-(m,r,m)=\sum_{t=1}^{m}tv+(r-1)(m+1)+(m+1)-\sum_{t=1}^{m-1}tv+(r-1)m+m=mv+r=B$, $h^*=(m+1,r,m+1)$ holds. This ends the proof of the lemma. □

In the following we give a specific solution formula to solve the problem $1|p_j=p;\ B;\ S-\mathrm{batch}|\sum C_j$.

Algorithm 8 (specific solution formula). Let $B=mv+r$. Then for n-orders batch sizing problem,

(1) $h^*=(m+1,r,m+1)=\sum_{t=1}^{m}tv+r(m+1)$ and $n=\lceil(n-h^*)/B\rceil B+(d,i,w)=aB+(d,i,w)$,

(2) the number of batches $k=a+d$,

(3) batch sizes

$$b_t=\begin{cases}B, & \text{for } t=1,\ldots,a;\\ (d-t+a)\,v+i-1, & \text{for } t=a+1,\ldots,a+d-w;\\ (d-t+a)\,v+i, & \text{for } t=a+d-w+1,\ldots,a+d,\end{cases}\tag{25}$$

where $\lceil x\rceil$ denotes the minimal nonnegative integer greater than or equal to rational number x.

Theorem 9. *Algorithm 8 give a specific calculating formula to solve problem* $1|p_j=p;\ B;\ S-batch|\sum C_j$ *with* $S=vp$.

Proof. By (ii) in Lemma 4 and $\mathrm{SUCC}(j)=j-B$ if $j\geq h^*$, along with $(n-h^*)/B\leq\lceil(n-h^*)/B\rceil<(n-h^*)/B+1$, we have that $n-(\lceil(n-h^*)/B\rceil-1)B>n-((n-h^*)/B+1-1)B=h^*$. This implies that $b_t=B$ for $t=\{1,\ldots,\lceil(n-h^*)/B\rceil\}=\{1,\ldots,a\}$ and $\mathrm{SUCC}(n-(\lceil(n-h^*)/B\rceil-1)B)=n-(\lceil(n-h^*)/B\rceil-1)B-B=n-(\lceil(n-h^*)/B\rceil)B=(d,i,w)$. By $n-(\lceil(n-h^*)/B\rceil)B\leq n-((n-h^*)/B)=h^*$ and (i) in Lemma 7, we have that $\mathrm{SUCC}(d,i,w)=(d-1,i,w),\ldots,\mathrm{SUCC}(w+1,i,w)=(w,i,w)$, and $\mathrm{SUCC}(w,i,w)=(w-1,i,w-1),\ldots,\mathrm{SUCC}(1,i,1)=0$. Thus, for $s=0,\ldots,d-w-1$, $(d-s,i,w)-(d-s-1,i,w)=(d-s-1)v+(i-1)$; that is, for $t=a+1,\ldots,a+d-w$, $b_t=(d-t+a)v+(i-1)$ and for $s=d-w,\ldots,d-1$, $(d-s,i,w)-(d-s-1,i,w-1)=(d-s-1)v+i$; that is, for $t=a+d-w+1,\ldots,a+d$, $b_t=(d-t+a)v+i$. This ends the proof of the theorem. □

5. Results and Discussion

In this section, we will present three numerical examples to show the effectiveness of our proposed algorithms.

Example 1. Solve by Algorithm 2 the following instance of the $1|p_j=p;\ B;\ S-\mathrm{batch}|\sum C_j$ problem:

$$\begin{aligned}n&:=9,\\ p&:=2,\\ S&:=5,\\ B&:=5.\end{aligned}\tag{26}$$

Solution. Let $F(j,i)=j[S+(j-i)p]+F(i)$. We have $F(j)=\min_{i=\max\{0,j-B\}}^{j-1}F(j,i)$. Set $F(0)=0$.

By $F(1,0)=1[5+(1-0)2]+F(0)=7$, we have that $F(1)=7$. Thus $\mathrm{SUCC}(1)=0$.

By $F(2,0)=2[5+(2-0)2]+F(0)=18$ and $F(2,1)=2[5+(2-1)2]+F(1)=21$, we have that $F(2)=18$. Thus $\mathrm{SUCC}(2)=0$.

By $F(3,0)=33$, $F(3,1)=34$, and $F(3,2)=39$, we have that $F(3)=33$. Thus $\mathrm{SUCC}(3)=0$.

By $F(4,0)=52$, $F(4,1)=51$, $F(4,2)=54$, and $F(4,3)=61$, we have that $F(4)=51$. Thus $\mathrm{SUCC}(4)=1$.

By $F(5,0)=75$, $F(5,1)=72$, $F(5,2)=73$, $F(5,3)=78$, and $F(5,4)=86$, we have that $F(5)=72$. Thus $\mathrm{SUCC}(5)=1$.

By $F(6,1)=97$, $F(6,2)=96$, $F(6,3)=99$, $F(6,4)=105$, and $F(6,5)=114$, we have that $F(6)=96$. Thus $\mathrm{SUCC}(6)=2$.

By $F(7,2)=123$, $F(7,3)=124$, $F(7,4)=128$, $F(7,5)=135$, and $F(7,6)=145$, we have that $F(7)=123$. Thus $\mathrm{SUCC}(7)=2$.

By $F(8,3)=153$, $F(8,4)=155$, $F(8,5)=160$, $F(8,6)=168$, and $F(8,7)=179$, we have that $F(8)=153$. Thus $\mathrm{SUCC}(8)=3$.

By $F(9,4)=186$, $F(9,5)=189$, $F(9,6)=195$, $F(9,7)=204$, and $F(9,8)=216$, we have that $F(9)=186$. Thus $\mathrm{SUCC}(9)=4$.

We obtain the optimal solution of the problem instance: the number of batches $k=3$; batch sizes $b_1=5$, $b_2=3$, and $b_3=1$.

Example 2. Solve by Algorithm 5 the following instance of the $1|p_j=p;\ B;\ S-\mathrm{batch}|\sum C_j$ problem:

$$\begin{aligned}n&:=81,\\ p&:=2,\\ S&:=5,\\ B&:=5.\end{aligned}\tag{27}$$

Solution. By doing same calculations in Example 1, we have that SUCC(1) = 0, SUCC(2) = 0, SUCC(3) = 0, SUCC(4) = 1, SUCC(5) = 1, SUCC(6) = 2, and SUCC(7) = 2. Noting the fact that $h^* = 7$ is the minimal integer such that SUCC(h^*) = t^* and $h^* - t^* = B$, along with Algorithm 5, we obtain the optimal solution of the problem instance: the number of batches $k = 17$; batch sizes $b_t = 5$ for $t = 1, \ldots, 15$, $b_{16} = 4$, and $b_{17} = 2$.

Example 3. Solve by Algorithm 8 the following instance of the $1|p_j = p;\ B;\ S - \text{batch}|\sum C_j$ problem:

$$\begin{aligned} n &:= 2000, \\ p &:= 7, \\ S &:= 28 = 4 \times p, \\ B &:= 111. \end{aligned} \tag{28}$$

Solution. Since $B = 27 \times 4 + 3$, we have that $h^* = (28, 3, 28) = \sum_{t=1}^{27} 4t + 3 \times 28 = 1596$. 2000 can be written as $2000 = \lceil (2000 - 1596)/111 \rceil 111 + (d, i, w) = 4 \times 111 + 1556 = 4 \times 111 + (27, 6, 17)$. By Algorithm 8, we obtain the optimal solution of the problem instance: the number of batches $k = 31$; batch sizes $b_t = 111$ for $t = 1, 2, 3, 4$, $b_t = 4(27 - t + 4) + 5 = 129 - 4t$ for $t = 5, \ldots, 14$, and $b_t = 4(27 - t + 4) + 6 = 130 - 4t$ for $t = 15, \ldots, 31$.

6. Conclusions

In this paper, we study the supply chain problem of batching identical orders with lifespan constraints and from the same location on a single machine. The problem is to choose the number of batches and batch sizes to minimize the total delivery time without violating the order lifespan. We first give a backward dynamic programming algorithm, but it is not an actually polynomial-time algorithm. Then, we do further research into the recursion formula in the dynamic programming algorithm and prove that $j - \text{SUCC}(j) = B$ when j is a sufficiently large integer. Following from the property, a constant time partial dynamic programming algorithm is given. Further, using the difference characteristics of the optimal value function, a specific calculating formula to solve the problem in case $S = vp$ is obtained. An interesting open question is whether there is a specific calculating formula to solve the problem $1|p_j = p;\ B;\ S - \text{batch}|\sum C_j$ in a general case of S. The difference analysis technique should be able to be used to study some other supply chain scheduling problem with lifespan constraints.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors would like to express their extremely gratefulness to the referee and editor for their useful comments and suggestions that improved this paper. This research was supported in part by the Zhejiang Natural Science Foundation of China Grant Y6110054 and the Hong Kong Polytechnic University Research Grant J-BB7D and The Hong Kong CERG Research Fund 5515/10H.

References

[1] C. D. Tarantilis and C. T. Kiranoudis, "A meta-heuristic algorithm for the efficient distribution of perishable foods," *Journal of Food Engineering*, vol. 50, no. 1, pp. 1–9, 2001.

[2] C. Arbib, D. Pacciarelli, and S. Smriglio, "A three-dimensional matching model for perishable production scheduling," *Discrete Applied Mathematics*, vol. 92, no. 1, pp. 1–15, 1999.

[3] H.-K. Chen, C.-F. Hsueh, and M.-S. Chang, "Production scheduling and vehicle routing with time windows for perishable food products," *Computers & Operations Research*, vol. 36, no. 7, pp. 2311–2319, 2009.

[4] P. Farahani, M. Grunow, and H.-O. Günther, "Integrated production and distribution planning for perishable food products," *Flexible Services and Manufacturing Journal*, vol. 24, no. 1, pp. 28–51, 2012.

[5] C. A. Méndez, G. P. Henning, and J. Cerdá, "Optimal scheduling of batch plants satisfying multiple product orders with different due-dates," *Computers and Chemical Engineering*, vol. 24, no. 9-10, pp. 2223–2245, 2000.

[6] P. Amorim, M. A. F. Belo-Filho, F. M. B. Toledo, C. Almeder, and B. Almada-Lobo, "Lot sizing versus batching in the production and distribution planning of perishable goods," *International Journal of Production Economics*, vol. 146, no. 1, pp. 208–218, 2013.

[7] B. Bilgen and H.-O. Günther, "Integrated production and distribution planning in the fast moving consumer goods industry: a block planning application," *OR Spectrum*, vol. 32, no. 4, pp. 927–955, 2010.

[8] S. Kaplan and G. Rabadi, "Exact and heuristic algorithms for the aerial refueling parallel machine scheduling problem with due date-to-deadline window and ready times," *Computers and Industrial Engineering*, vol. 62, no. 1, pp. 276–285, 2012.

[9] N. Shirvani, R. Ruiz, and S. Shadrokh, "Cyclic scheduling of perishable products in parallel machine with release dates, due dates and deadlines," *International Journal of Production Economics*, vol. 156, pp. 1–12, 2014.

[10] G. M. Kopanos, L. Puigjaner, and M. C. Georgiadis, "Simultaneous production and logistics operations planning in semicontinuous food industries," *Omega*, vol. 40, no. 5, pp. 634–650, 2012.

[11] B. Bilgen and Y. Çelebi, "Integrated production scheduling and distribution planning in dairy supply chain by hybrid modelling," *Annals of Operations Research*, vol. 211, no. 1, pp. 55–82, 2013.

[12] H. N. Geismar, G. Laporte, L. Lei, and C. Sriskandarajah, "The integrated production and transportation scheduling problem for a product with a short life span and noninstantaneous transportation time," *INFORMS Journal on Computing Informs*, vol. 20, no. 1, pp. 21–33, 2008.

[13] R. Armstrong, S. Gao, and L. Lei, "A zero-inventory production and distribution problem with a fixed customer sequence," *Annals of Operations Research*, vol. 159, pp. 395–414, 2008.

[14] C. Viergutz and S. Knust, "Integrated production and distribution scheduling with lifespan constraints," *Annals of Operations Research*, vol. 213, pp. 293–318, 2014.

[15] J. M. García, K. Smith, S. Lozano, and F. Guerrero, "A comparison of GRASP and an exact method for solving a production and delivery scheduling problem," in *Hybrid Information Systems*, vol. 14 of *Advances in Soft Computing*, pp. 431–447, Physica, Heidelberg, Germany, 2002.

[16] J. M. Garcia and S. Lozano, "Production and delivery scheduling problem with time windows," *Computers and Industrial Engineering*, vol. 48, no. 4, pp. 733–742, 2005.

[17] J. M. Garcia and S. Lozano, "Production and vehicle scheduling for ready-mix operations," *Computers and Industrial Engineering*, vol. 46, no. 4, pp. 803–816, 2004.

[18] S. Zangeneh-Khamooshi, Z. B. Zabinsky, and J. A. Heim, "A multi-shift vehicle routing problem with windows and cycle times," *Optimization Letters*, vol. 7, no. 6, pp. 1215–1225, 2013.

[19] M. G. van Buer, D. L. Woodruff, and R. T. Olson, "Solving the medium newspaper production/distribution problem," *European Journal of Operational Research*, vol. 115, no. 2, pp. 237–253, 1999.

[20] Z.-L. Chen, "Integrated production and outbound distribution scheduling: review and extensions," *Operations Research*, vol. 58, no. 1, pp. 130–148, 2010.